SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Voditelj rada:

Prof.dr.sc. Mario Essert

Božidar Štimac

ZAGREB, rujna 2012.

SADRŽAJ

Izjavljujem da sam ovaj rad izradio samostalno sa stečenim znanjem i navedenom literaturom.

Zahvaljujem se svom mentoru na vodstvu i korisnim savjetima.

1. UVOD

Ovaj rad ima za cilj istražiti mogućnosti automatskog označivanja informacije. Označivanje informacije ostvaruje se meta-tagovima (oznakama) među kojima prednjači XML (EXtensible Markup Language). Za takav rad razvijeni su različiti editori, mrežni i lokalni. Međutim sam proces označIvanja je dugotrajan, težak i podložan pogreškama, budući da se obavlja isključivo ručno. Automatsko označivanje bi ostavilo korisniku i dalje mogućnosti ručnih izmjena i standarnog načina rada, ali bi većina posla bila ostvarena uz pomoć jednom definiranih pojmova i njihovih prikladnih oznaka (tagova) ili opisa.

Za razvoj aplikacije za automatsko označivanje WEB sadržaja odabran je programski jezik javascript, koji se izvršava lokalno na korisnikovom računalu unutar WEB preglednika. Pokraj programskog jeika, potrebno je odabrati i WEB editor nad kojim će se razvijati spomenuta aplikacija.

WEB editori su aplikacije koje omogućavaju korisnicima jednostavno uređivanje tekstualnog sadržaja putem internet preglednika. Isto tako su prilagođeni za programere, kako bi ih jednostavno ugradili u razvoju vlastitih aplikacija.

Neki od poznatijih WEB editora su CKEditor i TinyMCE editor, a u sklopu ovog rada odabran je TinyMCE editor kao temeljna podloga za razvoj aplikacije za automatsko označivanje WEB sadržaja.

TinyMCE editor korisnicima omogućuje osnovne funkcije uređivanja teksta, a neke od tih funkcija su:

- veličina i vrsta slova
- podebljana, podcrtana ili nakošena slova
- boja slova ili boja pozadine slova

Sa simpatičnim sučeljem podsjeća na široko poznati Microsoft Word, te na taj način ne predstavlja problem za korištenje.

Slika 1: TInyMCE korisničko sučelje

S druge strane, TinyMCE editor omogućuje programerima pristup do unesenog sadržaja od strane korisnika, kako bi mogli dalje upravljati i kontrolirati taj sadržaj. Važno je napomenuti da osim vidljivog teksta unutar editora, postoje djelovi teksta koji su nevidljivi korisnicima. Ti dijelovi teksta su izdvojeni u trokutastim zagradama, nazivaju se oznake (tagovi) i u njima se nalaze informacije koje se

brinu o uređenju teksta. Npr. Riječ „TinyMCE“ na slici korisničkog sučelja editora ima podebljana slova „MCE“, što je ostvareno pomoću ovakvog ukupnog teksta:

** TinyMCE editor **

Takva struktura teksta se naziva XML .

XML je jezik za označivanje podataka. Ideja je bila stvoriti jedan jezik koji će biti jednostavno čitljiv i ljudima i računalnim programima. Princip realizacije je vrlo jednostavan, odgovarajući sadržaj je uokviren odgovarajućim oznakama koje ga opisuju i imaju poznato, ili lako shvatljivo značenje. Primjer strukture XML-a:

```
<knjižnica>
        <knjiga kategorija="roman">
                <naslov>Harry Potter</naslov>
                 <autor>J K. Rowling</autor>
                <godina>2005</godina>
                <cijena>29.99</cijena>
         </knjiga>
        <knjiga kategorija="WEB">
                <naslov>Learning XML</naslov>
                <autor>Erik T. Ray</autor>
                <godina>2003</godina>
                <cijena>39.95</cijena>
        </knjiga>
</knjižnica>
```

Oznake možemo promatrati kao matematičke zagrade, isto kao što postoji otvorena i zatvorena zagrada, tako postoji otvorena i zatvorena oznaka, koje skupa sa sadržajem kojeg označavaju čine jedan XML element. XML elemente također kao i zagrade možemo nizati ili ugnježdavati, a ne možemo križati. Za razliku od funkcije zagrada, XML element opisuje svoj sadržaj prvenstveno svojim nazivom koji se nalazi unutar oznaka omeđen trokutastim zagradama. XML element može biti opisan i sa atributima. U navedenom primjeru XML element naziva „knjiga“ ima atribut „kategorija“ , koji dodatno opisuje sadržaj toga XML elementa.

Mogućnosti poboljšanja TinyMCE editora ugradnjom automatizma i slojeva

TinyMCE editor je ograničen ručnim uređivanjem željenog sadržaja, dok se pojavljuje potreba da se proces automatizira pošto se iste radnje moraju izvršavati na velikom broju objekata. Npr. Potrebno je pronaći riječ „mehanika“ gdje god se pojavljuje u nekom opsežnom tekstu, što već u startu predstavlja problem. Problem postaje tim veći ukoliko svakoj pronađenoj riječi treba dopisati pripadajuće oznake u svrhu stvaranja željenog XML-a zbog toga što niti jedan WEB editor to ne omogućava. WEB editori mogu dodati samo predefinirane oznake riječima, koje najčešće nose samo informaciju o uređenju teksta (veličina slova, boja slova, naslov, odlomak...), te je nemoguće označivati sadržaj sa proizvoljnim oznakama, osim ručnim unošenjem istih.

Automatizam bi omogućio, osim automatskog pronalaska riječi, i automatsko unošenje oznaka koje će osigurati željeni stil pronađenim riječima ali i povezati stvorene oznake s unaprijed definiranom bazom oznaka gdje korisnik definira vlastite atribute koje želi dodjeliti pronađenim riječima. Jednom tako označen tekst ostavlja korisniku razne mogućnosti daljnjeg upravljanja tim sadržajem prema vlastitim potrebama, isto kao što je i razvoj aplikacije moguće usmjeriti da ispuni novononastale korisničke zahtjeve.

Pokraj automatskog označivanja teksta, poželjeno svojstvo označenog teksta je i mogućnost isključivanja željenih oznaka, ali ne i njihovo brisanje kako bi bilo moguće ponovno uključiti jednom već unešene oznake. Za to svojstvo je potrebno osigurati različite razine tj. slojeve na kojima bi korisnik mogao upravljati tekstom.

TinyMCE editor je alat koji se stalno razvija od strane vlastitog razvojnog tima i tako omogućuje uvijek nove funkcije koje olakšavaju rad programerima koji su se odlučili koristiti ovaj gotovi alat. Valja primjetiti da alat koji je izložen stalnom razvoju, također je i stalno u testnoj (beta) fazi koja je okarakterezirana mogućim nerješenim sitnim nepravilnostima u radu. Te nepravilnosti ne moraju predstavljati problem kod jednostavnog označavanja teksta, ali kod susreta s drugom programskom logikom koja zavisi od poznavanja točnih pozicija oznaka, greške mogu biti fatalne, tj. može doći do nemogućnosti daljnje obrade podataka i zamrzavanje rada programa. Zbog toga sam odlučio koristiti samo tri osnovne naredbe koje je omogućio TinyMCE razvojni tim, a to su naredbe:

- za dohvaćanje ukupnog teksta iz editora
- za dohvaćanje selektiranog teksta od strane korisnika
- za postavljanje željenog teksta natrag u editor

Osim prednosti ovakvog pristupa iz sigurnosnih razloga, prednost je i ta što razvijena aplikacija neće biti strogo ovisna o TinyMCE editoru, nego će se lagano moći prilagoditi bilo kojem WEB editoru.

2. TinyMCE WEB editor

1.1. Osnovni XML elemtni podržani unutar TinyMCE editora

Razvojni tim TinyMCE editora je odabrao koji XML elemnti i s kojim mogućim atributima se koriste unutar njihovog editora, kako bi omogućili funkcionalnost uređivanja teksta.

Osnovni XML elementi koji su podržane unutar TinyMCE editora su:

- -služi za definiranje odlomka
- -služi za označivanje određenog dijela teksta

Pomoću SPAN XML elementa je moguće zadati željeni stil označenom sadržaju . Kako bi SPAN XML element pohranio potrebnu informaciju o stilu teksta, potrebno mu je dodati željenu informaciju unutar atributa „style“, koja se se zapisuje sukladno pravilima HTML-a (HyperText Markup Language), jezika koji je također po svojoj strukturi XML i služi za izradu internet stranica. Sljedeći primjer pokazuje kako se može definirati boja slova i pozadinska boja :

Označeni tekst

1.2. Proširenje liste podržanih XML elemenata

Osim osnovnih XML elemenata koji se koriste unutar TinyMCE editora, programerima je omogućeno da prošire listu podržanih XML elemenata prema vlastitim potrebama. Za potrebe razvoja ovoga rada, lista podržanih XML elemenata je proširena sa dva nova elementa i njegovim pripadajućim atributima, čija će primjena biti objašnjena kasnije :

- <check></check>
- <tie id='' class=''></tie>

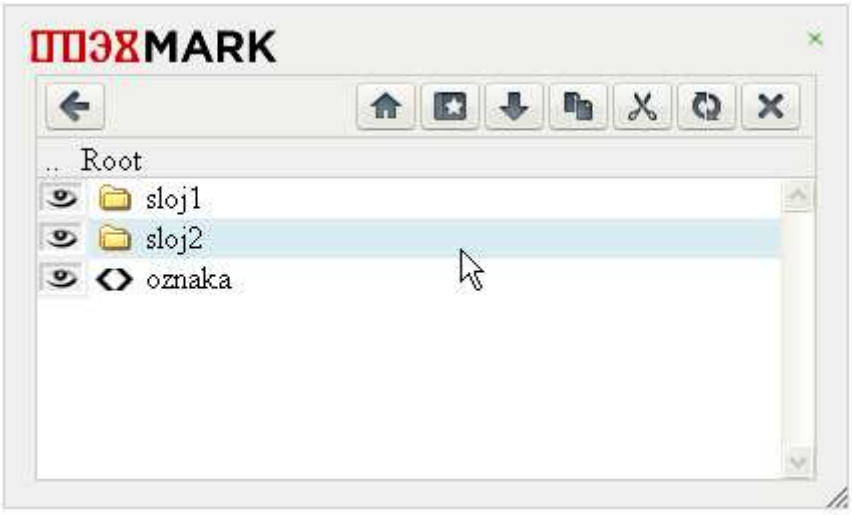
3. Slojevi i čvorovi

3.1. Svrha slojeva

Slojevi omogućavaju obradu istog teksta od strane jednog ili više korisnika, bez gubljenja niti jedne već stvorene informacije, dok s druge strane omogućuje bolju vizualnu preglednost i veću funkcionalnu fleksibilnost.

Korisnik ima bolju vizualnu preglednost jer dobiva mogućnost organizirati definirane oznake u različitim slojevima, isto kao što organiziramo datoteke unutar direktorija Microsoft Windows operativnog sustava. Analogno kao što je moguće stvarati direktorije unutar drugih direktorija, tako je moguće i stvarati slojeve unutar drugih slojeva, te su moguće i funkcije poput brisanja, kopiranja ili premještanja cijelih slojeva ili pojedinačnih oznaka. Važno je primjetiti da kad se u ovom kontekstu spominje riječ „oznaka“, to se onda odnosi na podatak koji nosi informaciju o definiranoj oznaci. Slično kao kad kažete da imate slike u direktoriju, a tamo zapravo slike ne postoje nego samo datoteke sa informacijom o slikama, koje može otvoriti samo sa programom koji je namjenjen za pregled slika.

Za razliku od direktorija, slojevi imaju svojstvo vidljivosti koje može biti uključeno ili isključeno. Ukoliko isključimo vidljivost pojedinom sloju, automatski smo isključili vidljivost svim slojevima i oznakama koje se nalaze unutar tog sloja.
Za upravljanje slojevima korisniku pomaže sljedeće sučelje:

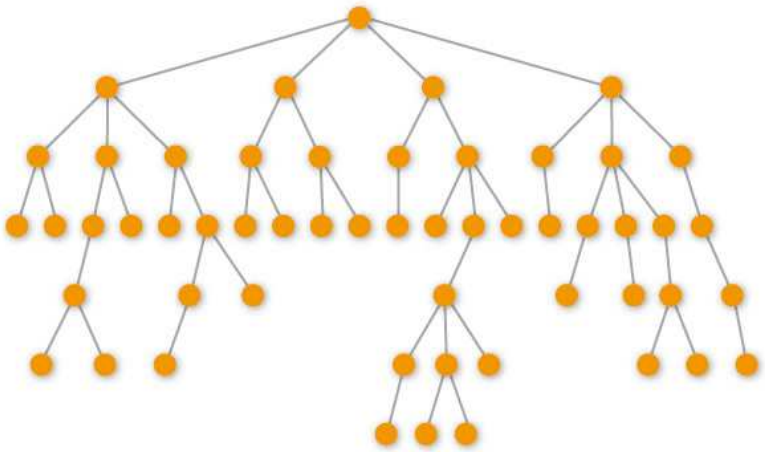


Slika 2: Sučelje za upravljanje slojevima i oznakama

Jednom kad je korisnik napravio željene slojeve i u njima definirao vlastite oznake, obrada teksta postaje trivijalna stvar. Dovoljno je odabrati željeni sloj ili pojedinačnu oznaku i djelovati s njima na tekst pomoću ponuđenih akcija, koje ovise i o samoj vrsti oznaka i načinu njihove primjene. Također funkcije koje se primjenjuju na već označeni tekst mogu ignorirat oznake koje se nalaze unutar slojeva čija je vidljivost isključena, na taj način je povećana korisnička fleksibilnost.

3.2. Organiziranje varijable za pohranu slojeva

Slojevi su stablena struktura koja se može prikazati grafom:



Slika 3: Stablena struktura slojeva

Ako se graf promatra odozgo prema dolje vidi se da se svaki čvor može granati na više čvorova, dok u suprotnom smjeru se vidi da svaki čvor ima vezu sa samo jednim čvorom. To svojstvo stablene strukture određuje da svaki sloj može imati samo jednog roditelja i neograničen broj djece. Pojedini sloj može imati djecu koja su također slojevi ili su oznake, dok oznaka može imati samo djecu koja su atributi dodjeljeni toj oznaci od strane korisnika prilikom stvaranja te oznake. Osim atributa koje je korisnik dodijelio željenoj oznaci, svaka oznaka ili sloj imaju atribute koji su pohranjeni unutar varijable koja opisuje taj čvor i koji su nužni za funkcionalnost programa. Glavni atribut kojeg imaju svi čvorovi i po kojem se razlikuju je „id“, to je ključ za dohvaćanje željenog čvora. Neki od ostalih atributa su:

- name –ime čvora
- tip –definira hoće li čvor biti sloj , oznaka ili dodjeljeni atribut oznake
- child- je lista svih id-ova djece pojedinog čvora
- childPosition-indeks pozicije čvora spram susjednih čvorova
- parent-je id roditelja pojedinog čvora
- visible -je dozvola za vidljivost sloja od strane nadređenih čvorova
- localVisible-vidljivost pojedinog sloja

Postoje i atributi koji ovdje nisu navedeni a koji se tiču samo nekih čvorova. Na primjer oznaka ima atribute koji definiraju njezin tip namjene, naziv XML elementa kojeg čini ta oznaka, opis ili osjetljivost oznake na velika i mala slova. Ali ti atributi trenutno nisu potrebni za razumjevanje odnosa među čvorovima.

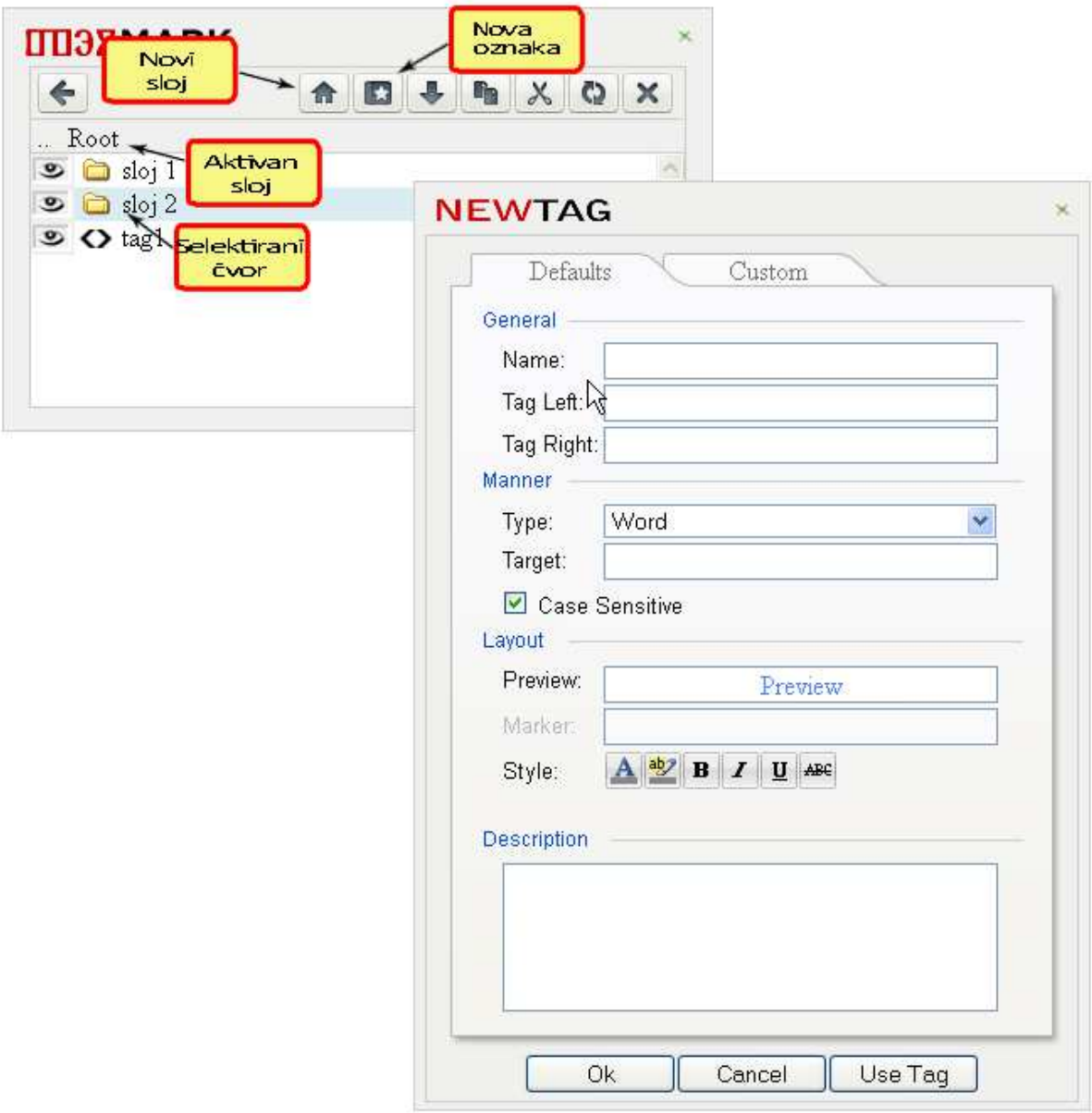
3.3. Stvaranje novog čvora

Stvaranje novog čvora znači stvaranje nove varijable koja je po svojoj prirodi objekt. Sukladno tome od kuda je pozvana funkcija za stvaranje novog čvora, tip čvora će biti automatski definiran. Isto tako moguće je odrediti atribut „parent“ jer je osigurana varijabla koja prati „id“ aktivanog sloja unutar izbornika za upravljanje slojevima, a korisnik je primoran prilikom stvaranja novog čvora nalaziti se unutar njegovog roditelja, tj. postaviti njegovog roditelja kao aktivan sloj.

Bitan dio je kako pohraniti novonastali čvor kako bi kasnije bio osiguran jednostavan pristup tom čvoru. Za to je stvorena globalna varijabla „alltag“ koja je po svojoj prirodi lista. Pri stvaranju svakog novog čvora, program provjerava duljinu liste „alltag“ i ovisno o toj duljini određuje vrijednost atributa „id“, te novonastali čvor se sprema na kraj liste „alltag“, tj. na poziciju koja se točno poklapa sa vrijednošću atributa „id“.

Nakon što je stvoren novi čvor, potrebno je umetnuti njegov „id“ u listu djece njegovog roditelja kako bi veze među čvorovima bile potpune.

S korisničke strane stvaranje novog čvora se izvršava putem sučelja za upravljanje slojevima:



Slika 4: Kreiranje novog čvora

Ukoliko korisnik stvara samo novi sloj, sve što će biti potrebno je odabrati ime novom sloju, a ukoliko stvara novu oznaku otvorit će mu se dijaloški okvir (kartica) za stvaranje nove oznake kao što je prikazano na slici. Kartica za stvaranje nove oznake sadrži dvije podkartice:

- Defaults – za definiranje osnovnih atributa
- Custom – za definiranje proizvoljnih atributa

Pomoću tih kartica korisnik može definirati neke atribute oznaka, redom to su:

- ime
- oznaka
- vrsta

- meta
- osjetljivost na velika slova
- marker
- stil
- opis

Navedeni atributi nose željenu informaciju o oznaci i njihova točna uloga i primjena na tekst će biti opisana kasnije.

Na drugoj podkartici korisnik stvara vlastite atribute i dodjeljuje im željenu vrijednost. Ti proizvoljni atributi su već opisani kao treća vrsta čvorova, tj. korisnički atributi oznaka.

Nakon što je korisnik kreirao novi sloj ili novu oznaku, taj novonastali čvor će se automatski stvoriti unutar aktivnog sloja i ispod selektiranog čvora. Ukoliko niti jedan čvor nije selektiran, stvorit će se na zadnjoj poziciji.

3.4. Funkcije za pristup pojedinim čvorovima

3.4.1. Osnovno dohvaćanje čvorova

Kao što je spomenuto u prethodnom poglavlju, najosnovnije dohvaćanje čvorova izvodimo pomoću njegovog atributa „id“. Dovoljno je pozvat objekt na toj poziciji unutar liste „alltag“:

```
var tag = alltag[id];
```

Sljedeći primjer pokazuje kako možemo pozvati roditelja čvora čiji nam je „id“ poznat:

```
var tagParent = alltag[alltag[id].parent];
```

Radi lakšeg razumijevanja možemo isti izraz rastaviti na faktore:

```
var tag = alltag[id];
var parentId = tag.parent;
var tagParent = alltag[parentId];
```

Sljedeći primjer pokazuje kako možemo pozvati djete čvora čiji nam je „id“ poznat, ali nam mora biti poznat i indeks pozicije djeteta:

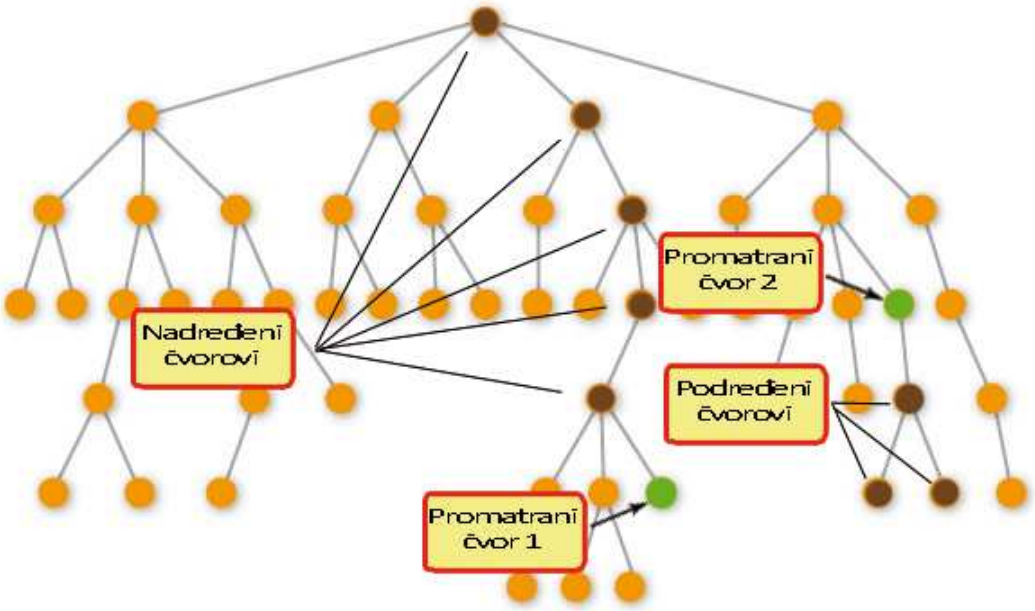
```
var tagChild = alltag[alltag[id].child[index]];
```

Ukoliko želimo dohvatiti listu braće pojedinog čvora tj. listu svih susjednih čvorova, dovoljno je pozvati listu djece od roditelja tog čvora:

```
var brothers = alltag[alltag[id].parent].child;
```

3.4.2. Dohvaćanje svih nadređenih čvora

Nakon dohvaćanja čvorova povezanih osnovnim vezama gdje nije bilo potrebno izgrađivati posebne funkcije, postoje učestala dohvaćanja koja zahtijevaju izgradnju rekurzivnih funkcija. Jedna od tih potreba je dohvaćanje svih nadređenih čvorova. To su svi čvorovi koji se nalaze na liniji stabla krečući se od promatranog čvora prema gore sve do prvog čvora.



Slika 5: Nadređeni i podređeni čvorovi

```
function lookParents(tag,parents)  //kod prvog poziva se izostavi parents
      {
      if(!parents)var parents=[];
      parents.push(tag.parent);
      if(tag.parent!=0) parents=lookParents(alltag[tag.parent],parents);
      return parents;
      }
```

Funkcija „lookParents()“ se poziva samo s jednim argumentom, koji je promatrani čvor čije roditelje želimo pronaći. Kod prvog poziva funkcija prepoznaje da nedostaje drugi argument, te stvara praznu listu u koju će pohranjivati sve id-ove roditelja promatranog čvora. Nakon što je stvorena potrebna lista i u nju pohranjen roditelj promatranog čvora, funkcija će pozvati sama sebe a kao argumente će poslati upravo roditelja promatranog čvora i listu u koju se spremaju id-ovi roditelja. Na taj način je ostvareno da funkcija poziva sama sebe sve dok ne dodje do prvog stvorenog čvora, kojeg će prepoznati po vrijednosti atributa „id“ i na taj način završiti sa rekurzijom.

3.4.3. Dohvaćanje svih podređenih čvorova

Funkcija za dohvaćanje svih podređenih čvorova funkcionira analogno funkciji za dohvaćanje svih nadređenih čvorova. Jedina razlika je u tome što za razliku od roditelja, čvor može imati više od jednog djeteta, te zbog toga je nužno osigurati petlju koja će napraviti rekurzivni poziv za svako dijete promatranog čvora.

```
function lookChilds(tag,childs)//kod prvog poziva se izostavi childs
	{
	if(!childs)var childs=[];
	for(var x in tag.child)
		{
		childs.push(tag.child[x]);
		childs=lookChilds(alltag[tag.child[x]],childs)
		}
	return childs
	}
```

3.4.4. Dohvaćanje čvorova prema zadanim atributima

Do sada sve funkcije za pristup čvorovima su se temeljile na vezama među čvorovima. Za razliku od takvih funkcija potrebno je imati sasvim slobodan pristup bilo kojem čvoru u ovisnosti koje informacije želimo pronaći unutar čvora. Funkcije s takvim pristupom se temelje na usporedbi željenih vrijednosti atributa koje tražimo i stvarnih vrijednosti atributa unutar čvorova.
Sljedeća funkcija omougćuje pretragu unutar izdvojene liste čvorova ili ukupne liste čvorova „alltag“, te je moguća pretraga sa prozvoljnim brojem atributa i pripadajućih vrijednosti koje želimo imati unutar traženih čvorova:

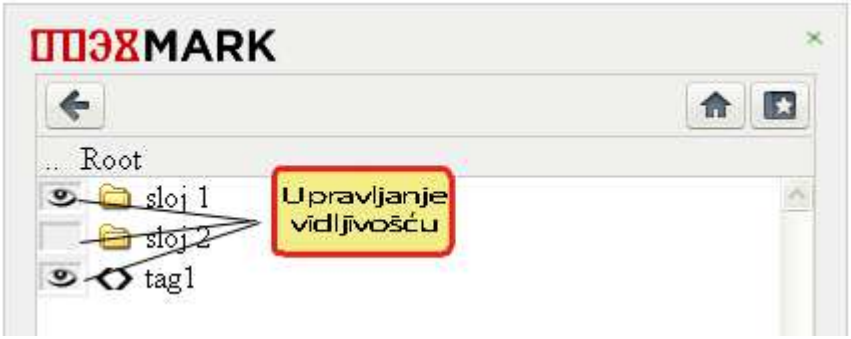
```
function findBy()  //argumenti moraju izgledati redom:
                   //(list,[atribut,value],[atribut2,value2]..)
                   //list je skupina id-ova u kojima se vrsi pretraga
                   //ostali argumenti su porizvoljne duljine
                   //vraca listu id-ova koji zadovoljavaju pretragu
	{
	var list=[];
	if(arguments[0]==alltag) for(var x=0;x<alltag.length;x++)list.push(x)
	else list=arguments[0];
	var list2=list.slice();
	for(var z=1;z<arguments.length;z++)
		{
		list=list2.slice();
		list2=[];
		for(var x in list)
			{
			if(alltag[list[x]][arguments[z][0]])
			{
		// !!! l() je funkcija koja pretvara string u mala slova
		if(l(alltag[list[x]][arguments[z][0]])==l(arguments[z][1]))
                                {
                                list2.push(alltag[list[x]].id);
                                }
                        }
                }
        }
	return list2;
	}
```

3.5. Upravljanje vidljivošću slojeva

Vidljivost slojeva se temelji na već ranije spomnutim atributima pojedinog čvora:

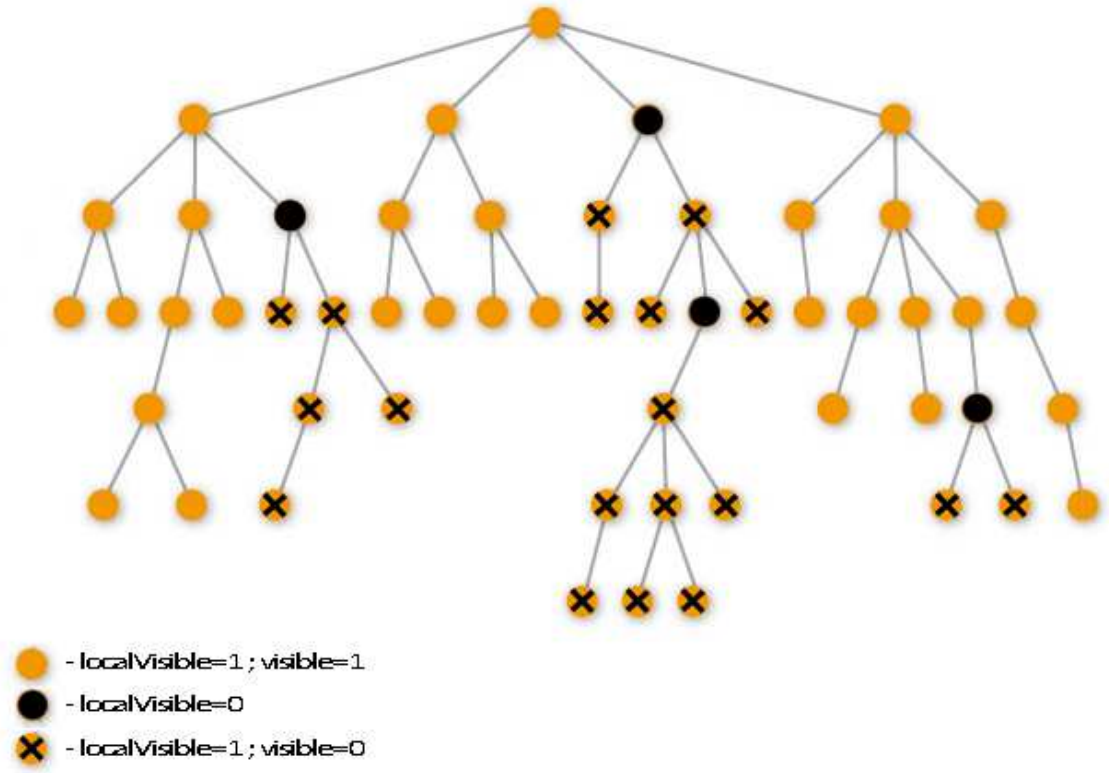
- visible
- localVisible

Ti atributi mogu nositi samo vrijednosti istine ili laži. Da bi pojedini sloj bio definiran kao vidljiv, mora imati obadva atributa postavljena u vrijednost istine. Korisnik upravlja atributom „localVisible“ klikom na sliku oka pored samog imena čvora, gdje može upaliti ili ugasiti vidljivost tom čvoru.



Slika 6: Upravljanje vidljivošću

Programski je to popraćeno prebacivanjem vrijednosti atributa „localVisible“ iz stanja istine u laž ili obratno, ali to nije dovoljno. Prilikom gašenja vidljivosti pojednim čvoru, korisnik zapravo kao posljedicu želi ugašenu vidljivost svim ostalim čvorovima koji se nalaze unutar toga čvora, bez obzira na njihovu vlastitu postavku vidljivosti. Dok kod ponovnog uključivanja vidljivosti tom istom čvoru, korisnik želi uključiti vidljivost samo onim čvorovima koji su u vlastitim postavkama definirani kao vidljivi. Atribut „localVisible“ je upravo taj koji definira samo vlastite postavke pojedinog čvora, dok atribut „visible“ definira dozvolu vidljivosti pojednom čvoru od strane nadređenih čvorova. Znači dovoljno je da samo jedan nadređeni čvor ima postavljen atribut „localVIsible“ kao laž, i sva njegova djeca moraju postaviti vrijednost atributa „visible“ u stanje laži, dok obratno ne vrijedi. Sljedeća slika prikazuje utjecaj vidljivosti pojedinog čvora na vidljivost ostalih čvorova.



Slika 7: Utjecaj vidljivosti između čvorova

Za upravljanje vidljivošću čvorova, također je konstruirana rekurzivna funkcija koja se aktivira pri promjeni vidljivosti željenog čvora koji je upravo prvi zadani argument funkcije.

```
function applyVisibleToFirstChilds(tag)
	{
	var childs=tag.child;
	for(var x in childs)
		{
		var visibleState=alltag[childs[x]].visible;
		alltag[childs[x]].visible=(tag.localVisible && tag.visible);
		var tagVisible= alltag[childs[x]].visible;
		var tagLocalVisible= alltag[childs[x]].localVisible;
		if(visibleState!=tagVisible && tagLocalVisible==1)
			{
			applyVisibleToFirstChilds(alltag[childs[x]])
			}
		}
	return;
	}
```

Funkcija „applyVisibleToFirstChilds()“ je prikazana pojednostavljeno, pošto u orginalnoj izvedbi ima dodatne linije koda koje izvršavaju određene operacije nad tekstom prilikom mijenjanja stanja vidljivosti pojedinog čvora, što trenutno nije bitno razmatrati. Zanimljivo je primjetiti da ovako konstruirana funkcija je optimizirana u smislu da neće dolaziti do onih čvorova za koje je očigledno da im nadređeni čvor nije mogao utjecati na stanje vidljivosti.

Prvi razlog koji prekida nepotrebnu rekurziju je provjera lokalne vidljivosti ispitivanog čvora. Ukoliko čvor nema lokalnu vidljivost, to je već dovoljan uvjet da sva njegova djeca nemogu imati dopuštenje za vidljivost i prema tome nema smisla nastavljati rekurziju za djecu toga čvora, pošto se njihovo stanje sigurno neće mijenjati.

Drugi razlog se temelji na varijabli „visibleState“ koja pamti stanje vidljivosti čvora prije utjecaja same funkcije na njega. Ako se uspostavi da se stanje tog čvora nije promijenilo, također nema smisla produbljivati rekurziju unutar toga čvora, jer ako nadređeni čvor nije mogao utjecati na ovaj čvor, nemože nikako utjecati niti na njegovu djecu.

Ova optimizacija može izgledati nepotrebna , ali treba se uzet u obzir da paralelno sa utjecajem na vidljivost pojedinog čvora se vrše određene operacije nad tekstom, koje prestaju biti zanemarive onog trenutka kad tekst postane dovoljno dugačak.

3.6. Kopiranje, premještanje i brisanje čvorova

Kopiranje, premještanje i brisanje čvorova su funkcije koje je nužno osigurati korisniku radi lakšeg organiziranja oznaka. S obzirom da je sad već poznato kako je organizirana varijabla za pohranu čvorova, moguće je razmotriti što se dešava s istom prilikom navedenih operacija.

Mnogi programeri su se susreli s problemom kopiranja ugnježdenih listi, i na prvu se može činiti da je sličan problem i ovdje kod kopiranja čvorova, ali zapravo je jedna bitna razlika. Za dobivanje novog čvora identičnog vlastitoj instanci je stvarno potrebno kopirati većinu atributa tog objekta, ali potrebno je istovremeno ponoviti proces stvaranja novog čvora, kako bi kopirani čvor imao vlastiti id, i time bio neovisan o svojoj instanci. Nadalje je potrebno promjeniti atribut „parent“ prema odgovarajućoj destinaciji gdje je čvor kopiran i također tom roditelju promijeniti atribut „childs“, te zadnji također jako bitan korak je ponoviti prva dva navedena koraka za svu djecu koja se nalaze unutar toga čvora, što opet dovodi do rekurzije. Ukoliko nebi ponovili navedene korake za svu djecu čvora kojeg želimo kopirati, desilo bi se to da kopirani čvor ne sadrži vlastitu djecu nego samo indeksira na djecu vlastite instance. Takvo ponašanje čak nije niti strano i poznato je pod nazivom „shortcut“ unutar Windows operativnog sustava.

Za razliku od kopiranja, premještanje je daleko jednostavnije. Dovoljno je utjecat na veze među čvorovima. Novoj i staroj destinaciji promijenit atribut „childs“ i premještenom čvoru promijenit atribut „parent“ sukladno željenoj destinaciji.

Kod brisanja čvorova je bitno paziti na duljinu liste „alltag“, pošto je ta duljina u strogom odnosu sa id-ovima čvorova. Dovoljno je zamijeniti čvor koji želimo obrisati sa praznim stringom kako se duljina nebi poremetila te postupak ponoviti i za svu djecu tog čvora.

3.7. Spremanje čvorova u bazu podataka

Spremanje čvorova je osigurano pomoću XML-a. Jednostavna funkcija pretvara atribute pojedinog čvora u string koji je valjan XML i kojega je kasnije jednostavno pročitati.

```
function createTagXmlString(tag)
      {
      var txt='<tag>';
      for(var x in tag)
            {
            if(x!='child') txt+='<'+x+'>'+tag[x]+'</'+x+'>'
            }
      txt+='</tag>';
      return txt;
   }
```

Kod stvaranja XML-a, neki atributi čak niti nisu potrebni za kompletan opis čvorova, kao što je npr. atribut „childs“. Ukoliko je pohranjena informacija o roditelju svakog čvora, moguće je saznati i suprotne veze koje indeksiraju na djecu čvorova, te je moguće prilikom učitavanja obnoviti te veze, tj. stovriti nove atribute „childs“. Obnavljanje veza čvorova prilikom učitavanja je u jednu ruku poželjno, jer je postojala potreba osigurati korisnicima mogućnost selektivnog učitavanja pohranjenih čvorova. Selektivno učitavanje omogućava korisnicima spajanje više različitih pohranjenih baza čvorova u jedan jedinstveni skup, što bi bez obnavljanja veza dovelo do dupliciranja id-ova i neispravnih veza među čvorovima.

4. Temeljne funkcije upravljanja označenim tekstom

4.1. Pozicioniranje unutar XML elementa

Pozicioniranje unutar XML elementa se sastoji od četiri funkcije čija kombinacija omogućuje pronalazak početka i završetka bilo otvorenog ili zatvorenog oznaka i to neovisno dali je poznata početna pozicija unutar otvorene ili zatvorene oznake. Redom to su:

- prvaZagrada(index,string)
- drugaZagrada(index,string)
- zagradaTagaZatvaranja(index,string)
- zagradaTagaOtvaranja(index,string)

Funkcije „prvaZagrada()“ i „drugaZagrada()“ primaju dva argumenta. Prvi argument je bilo koji indeks pozicije unutar oznake kojoj tražimo indeks pozicije prve ili druge zagrade, svejedno za otvorenu ili zatvorenu oznaku. Drugi argument je string u kojem se ta oznaka nalazi.

Funkcija „zagradaTagaZatvaranja()“ prima također dva argumenta, ali indeks pozicije se mora nalaziti unutar otvorene oznake kojoj tražimo poziciju prve zagrade pripadajuće zatvorene oznake. Treba biti svjestan da između otvorene i zatvorene oznake mogu postojati dodatni ugnježđeni XML elementi, koje funkcija mora izbjeći i pronaći odgovarajuću zatvorenu oznaku. To je ostvareno prebrojavanjem otvorenih i zatvorenih oznaka s desne strane promatrane otvorene oznake. Ukoliko je zadani string ispravan XML, svako pojavljivanje nove otvorene oznake mora rezultirati i pojavljivanjem jedne zatvorene oznake. Tek kad se te dvije vrijednosti izjednače, prvo pojavljivanje zatvorene oznake je tražena pozicija.

Analogno prethodnoj funkciji se ponaša i funkcija „zagradaTagaOtvaranja()“, samo se ovoj funkciji zadaje indeks pozcije unutar zatvorene oznake kojoj tražimo poziciju prve zagrade odgovarajuće otvorene oznake.

4.2. Brisanje oznaka

Pod brisanje oznaka se sporazumjeva oslobađanje određenog XML elementa od pripadajućih oznaka. Za brisanje oznaka iz zadanog stringa pomažu već kreirane funkcije za pozicioniranje unutar XML elementa. Ukoliko je poznat string u kojem se nalazi XML element i bilo koji indeks pozicije unutar oznaka koje želimo obrisati, možemo pronaći četiri kritične točke:

1. prva zagrada otvorene oznake
2. druga zagrada otvorene oznake
3. prva zagrada zatvorene oznake
4. druga zagrada zatvorene oznake

```
var string='primjer <tag>oznacenog</tag> sadrzaja'
```

Ako je poznata pozicija unutar otvorene oznake, na sljedeći način se određuju četiri kritične točke:

```
var t1=prvaZagrada(index,string);

var t2=drugaZagrada(index,string);

var t3=zagradaTagaZatvaranja(index,string);

var t4=drugaZagrada(t3,string);
```

U odnosu na kritične točke treba izdvojiti potrebne podstringove i spojiti ih kako bi dobili željeni string sa uklonjenim oznakama.

```
var string2=string[0,t1]+string[t2+1,t3]+string[t4+1,end];
```

4.3. Preskakanje oznaka

Funkcije za preskakanje oznaka nemaju samostalnu konačnu primjenu, ali su zato jedan od temelja koje koriste mnoge druge funkcije.

Tekst koji se prikazuje unutar editora ima čitljiv i jasan sadržaj, dok je u pozadini izmješan s raznim oznakama. Za preskakanje oznaka su izgrađene posebne funkcije koje omogućavaju preskakanje oznaka u desno, preskakanje oznaka u lijevo i preskakanje zatvorenih oznaka u lijevo, redom to su:

- filterTagR(indeks, string)
- filterTagL(indeks, string)
- filterTagLOnlyClosed(indeks, string)

Sve te funkcije rade na principu da primaju kao argumente string u kojem se očekuju oznake koje želimo zaobići i određena pozicija na kojoj se možda nalazi prva zagrada neke oznake. Funkcija provjerava vrijednost stringa na zadanoj poziciji dali odgovara znaku otvorene trokutaste zagrade. Ukoliko ne, vraća istu poziciju, a u suprotnom vraća prvu poziciju koja se nalazi izvan oznake, s tim da automatski preskače sve oznake koji su se nalazili u nizu.

4.4. Pretraga unutar označenog teksta

Pretraga unutar određenog stringa se inače može raditi pomoću ugrađenih funkcija nekog programskog jezika, dok kod označenog teksta to nikako nije slučaj jer se radi o pretragama koje korisnik izvršava nad vidljivim tekstom dok se traženi pojam mora pronaći unutar stringa koji je izmješan s oznakama. Korisnik može tražiti pojam „Hello world“ koji zapravo unutar stringa izgleda:

„He**ll****o world“**

Iz perspektive pretrage, broj i raspored oznaka nikako ne može biti poznat, te je potrebno napraviti tražilicu koja će biti neovisna o oznakama.

Za izradu takve tražilice prvo je potrebno napraviti običnu vlastitu tražilicu koju će biti moguće prilagoditi za pretragu unutar označenog teksta. Najednostavniji pristup izradi tražilice je zavrtiti petlju kroz string nad kojim se vrši pretraga te uspoređivati svaki znak sa prvim znakom pojma kojeg tražimo. Ako se prvi znak podudara, usporediti i drugi znak sa sljedećim znakom unutar stringa. I tako sve dok se znakovi ne raziđu ili dok se ne usporedi i zadnji znak unutar traženog pojma sa odgovarajućim znakom unutar stringa. Takav pristup bi funkcionirao u većini slučajeva ali skriva jedno malo iznanađenje. Onog trenutka kad usporedba dvaju znakova ne ispadne istinita, postoji mogućnost da je petlja već prošla točku u kojoj se nalazi pravi početak traženog pojma. Sljedeća slika prikazuje takav slučaj ukoliko je traženi pojam „ananas“ a string nad kojim se vrši pretraga „bananananastracciatella“ :

Slika 8

Slika prikazuje tri moguća početka riječi ananas koji se nalaze na indeksima 1,3 i 5, te je vidljivo da zadnji od njih predstavlja pravi početak riječi ananas. Prethodno opisana tražilica nebi prepoznala moguće početke na indeksima 3 i 5 jer bi bila zauzeta provjerom prvog mogućeg početka za kojeg bi tek na poziciji 6 zaključila da je neispravan jer je na toj poziciji očekivan znak „s“. Rješenje tog problema je moguće izvesti na više načina. Jedan način je da se nakon neuspjele usporedbe znakova, pretraga vraća na jednu poziciju iza zadnjeg provjeravanog mogućeg početka, što je vrlo jednostavno rješenje ali izaziva dodatne korake te ponavljanje već izvršenih usporedbi. Drugi pristup je nešto sofisticiraniji ali ne zahtjeva nikada negativne korake petlje. Taj pristup se temelji na pamćenju pozicije svakog mogućeg početka, te se na svakom koraku petlje ne vrši samo jedna usporedba nego po jedna usporedba za svaki spremljeni mogući početak, plus jedna dodatna usporedba za slučaj potrebnog stvaranja novog mogućeg početka. Usporedbe koje se odnose na pojedini mogući početak utječu dali će se taj mogući početak dalje pamtiti kao potencijalno pravi mogući početak ili brisati ako usporedba nije zadovoljavajuća.

Uvijet zadovoljavanja mogućeg početka za određeni korak petlje glasi:

```
if(string[x]==target[x-mp]){}
```

gdje su:

```
var x; //korak petlje
```

```
var target=ananas; //pojam koji se traži
```

```
var string='bananananastracciatella'; //string nad kojim se vrsi pretraga

var mp;// pozicija mogućeg početka pojma unutar stringa
```

Ukoliko pojedini mogući početak ispuni uvijet zadovoljavanja za određeni korak petlje, potrebno je provjeriti uvijet pronalaska:

```
if((x-mp)==target.length-1){}
```

Ukoliko je uvijet pronalaska istinit za određeni mogući početak, isti se vraća kao pozicija pronalaska traženog pojma i time je završena vlastita tražilica koja još nema mogućnost pretrage unutar označenog teksta. Za dodavanje tog svojstva koristit ćemo se funkcijom opisanom u prethodnom poglavlju za preskanje oznaka. U svakom koraku petlje potrebno je pozvati:

```
x=filterTagR(x, string);
```

Funkcija za preskakanje će prepoznati nailazak na prvu zagradu oznake te u tom slučaju preskočiti oznake tako što će podići vrijednost varijabli *x* na vrijednost indeksa prve pozicije koja se nalazi izvan oznaka, ili u suprotnom neće mijenjati vrijednost *x* varijable. Bitno je primjetiti da takvim utjecanjem na vrijednost varijable *x* se automatski remeti matematička korelacija indeksa unutar uvjeta zadovoljavanja određenog mogućeg početka i uvjeta pronalaska. Da bi se ta dva uvjeta i dalje mogla izvršavati, potrebno je uz svaki mogući početak pamtiti i koliko se pozicija nakon toga početka odnosilo na oznake, tj. koliko pozicija je preskočeno. Za takvu izvedbu je potrebno spremati moguće početke kao objekte s dva atributa, jedan koji govori o poziciji mogućeg početka a drugi o broju pozicija oznaka nakon tog mogućeg početka.

```
var mp=new Object();// varijabla koja pamti određeni mogući početak

mp.poz;// pozicija mogućeg početka pojma unutar stringa

mp.gap=0;//broj pozicija tagova nakon određenog mogućeg početka
```

Sada je moguće u sklopu funkcije za preskakanje oznaka, pamtiti preskočenu vrijednost unutar objekta mogućeg početka:

```
var xBefore=x;

x=filterTagR(x, string);

mp.gap+=x-xBefore;
```

Te je također potrebno prilagoditi uvjet zadovoljavanja određenog mogućeg početka:

```
if(string[x]==target[x-(mp.poz+mp.gap)]){}
```

i uvijet pronalaska:

```
if((x-(mp.poz+mp.gap))==target.length-1){}
```

4.5. Pomicanje kroz tekst neovisno o broju pozicija oznaka

Pomicanje kroz tekst neovisno o broju pozicija oznaka se također temelji na funkcijama za preskakanje oznaka. Postoji potreba za kretanjem kroz tekst za određeni broj koraka koji ne smiju biti ometani brojem pozicija koje zauzimaju oznake koje se nađu na putu. Za svaki pomaknuti indeks u željenom smjeru je potrebno pozvati odgovarajuću funkciju za preskakanje oznaka, kako konačna pozicija nebi završila unutar oznaka. Postoje dvije funkcije, za pomicanje u desno i za pomicanje u lijevo:

- moveTroughTagsR(index,steps,string)
- moveTroughTagsL(index,steps,string)

Funkcije primaju kao argumente indeks početne pozicije, broj koraka pomicanja i string u kojem se vrši pomicanje.

5. Označivanje teksta TIE oznakama

5.1. Svrha TIE oznaka

Za bolje razumjevanje TIE oznaka, prvo se treba osvrnuti na glavni problem označivanja unutar različitih slojeva. Već je spomenuto da gašenjem pojedinog sloja označene riječi gube pripadajući stil, što znači da unutar sadržaja editora je nužno i obrisati oznake koje su osiguravali taj stil. Ali brisanje tih oznaka samo zbog pripadajućeg stila bi značilo i gubljenje informacije o označenim riječima, prema tome ponovnim uključivanjem istog sloja nebi bilo moguće znati kojim riječima treba vratiti oznake. TIE oznake su upravo ta veza između određenih pozicija u tekstu i definiranih svojstava oznaka u određenim čvorovima.

TIE oznake ne obuhvaćaju nikakav sadržaj i na taj način nemaju nikakav utjecaj na vidljiv tekst, te nema opasnosti da ih korisnik slučajno obriše. Svaki TIE element ima dva atributa, jedan je „id“ koji je direktna veza sa id-om čvora kojeg predstavlja. Vrijednost toga atributa je jedinstveni ključ koji predstavlja neki čvor, ali pošto se isti čvor može koristiti za označivanje više pojmova nekog teksta, taj id se onda nikako ne može smatrati i jedinstvenim ključem nekog TIE elementa. Zato služi drugi atribut „class“, koji u ovom slučaju predstavlja jedinstveni ključ svakog TIE elementa. Osim atributa, svaki TIE element ima i pripadajući SPAN element čiji je sadržaj upravo onaj dio teksta kojeg je korisnik označio. Glavna uloga tog SPAN elementa je da osigura željeni stil označenom pojmu kako bi korisnik mogao vidjeti koje pojmove je označio i koje oznake su aktivne, međuostalim taj SPAN element omogućuje aktivaciju određenih funkcija prilikom klika mišom na označeni pojam. Važno je imati na umu da sva svojstva koja su zapisana unutar tog SPAN elementa dolaze upravo iz TIE elementa koji vadi tu informaciju iz varijable *alltag* gdje su pohranjeni svi čvorovi i pripadajući atributi, tj. svojstva koja je definirao korisnik.

Sljedeci primjer pokazuje raspored TIE elementa i pripadajuceg SPAN elementa:

Primjer **<tie id="ttie19" class="2"></tie>**oznacenog**** teksta.

Iz primjera se vidi da i SPAN element ima atribute „id“ i „class“,samo u ovom slučaju atribut „id“ stvarno predstavlja jedinstveni ključ tog elementa i ako se zanemari prefiks „stie“, odgovara upravo atributu „class“ unutar pripadajućeg TIE elementa, pomoću čega je stvorena čvrsta veza između ova dva elementa. Atribut „class“ predstavlja vezu sa id-om odgovarajućeg čvora, te onda naravno odgovara i atributu „id“ pripadajućeg TIE elementa.

5.2. Umetanje TIE oznaka i pripadajućeg stila

TIE oznake se umeću točno na poziciju gdje započinje označeni pojam od strane korisnika, prema tome samo TIE element je nedovoljan da svojom pozicijom odaje informaciju o indeksu pozicije gdje označeni pojam završava. Ta informacija je zapisana u samom čvoru koji se koristi za označivanje nekoga pojma i dohvaća se putem veze između TIE oznake i odgovarajućeg čvora, kao i svi ostali podaci koji se odnose na označeni pojam.

Prilikom označivanja, većina informacija nije potrebna za stvaranje TIE elementa i pripadajućeg SPAN elementa, ali dok god postoji čvrsta veza između TIE elementa i odgovarajućeg čvora, sve informacije su uvijek prisutne. Zbog te mogućnosti jednom označeni tekst pomoću TIE oznaka je podložan prilagodbi i nadogradnji raznih funkcija koje će različito reagirati na TIE elemente. Ukoliko naprimjer nastane potreba za dodatnim informacijama koje će nositi pojedini čvor, ista struktura teksta s TIE elementima će biti jednako dobra za realizaciju funkcija koje će ovisiti o tim novim informacijama.

Označivanje teksta je proces koji se vrši u dva glavna koraka. Prvi korak je umetanje TIE oznake na odgovarajuću poziciju u tekstu koja mora odgovarati početku pojma kojeg korisnik označuje. Naravno početak pojma ne treba shvatiti doslovno, jer TIE elementi se mogu nizati jedan na drugi a da si međusobno ne smetaju, što je omogućeno pomoću temeljnih funkcija za upravljanje označenim tekstom. Nakon pronalaska bilo kojeg TIE elementa unutar teksta, dovoljno se pozicionirati na njegovu prvu zagradu i zatim pozvati funkciju za preskakanje oznaka kako bi se konačno pozicionirali na početak pojma na kojeg se taj TIE element odnosi. U tom trenutku se aktivira funkcija „checkTIEvalid()“ koja provjerava dali taj pojam odgovara željenom objektu (meti) označivanja. Ukoliko odgovara, pronalazi završetak tog pojma i na taj način su određene potrebne pozicije za umetanje SPAN elementa. Drugi korak se sastoji od umetanja tog SPAN elementa u ovisnosti o pronađenim pozicijama. Tvorba samih SPAN oznaka ovisi o tome kako je korisnik definirao stil unutar čvora kojeg koristi za označivanje, a pronađene pozicije u najjednostavnijem slučaju su i konačne pozicije za umetanje SPAN oznaka. Pod najjednostavnijim slučajem se smatra da SPAN element se ne križa niti sa jednim drugim XML elementom unutar teksta.

Struktura teksta s TIE elementima omogućuje čak i označivanje onih pojmova koji se međusobno križaju. To je moguće zato jer TIE element nema nikakav sadržaj, te tako ne može doći u konflikt s nekim drugim TIE elementom, ali je zato nemoguće da njegov pripadajući SPAN element se križa sa nekim drugim SPAN elementom. Taj problem je rješen stvaranjem novog SPAN elementa koji označuje samo onaj dio teksta koji se nalazi unutar križanja, dok se oznake originalnih SPAN elemenata pomiču izvan granica križanja. Taj novi SPAN element svojim prepoznatljivim stilom crvene boje pozadine asocira korisnika da mu se elementi križaju. Važno je primjetiti da iako je moguće označiti tekst sa elementima koji se križaju, i dalje je nemoguće takav tekst pretvoriti u ispravan XML.

5.3. Brisanje TIE elementa i pripadajućeg stila

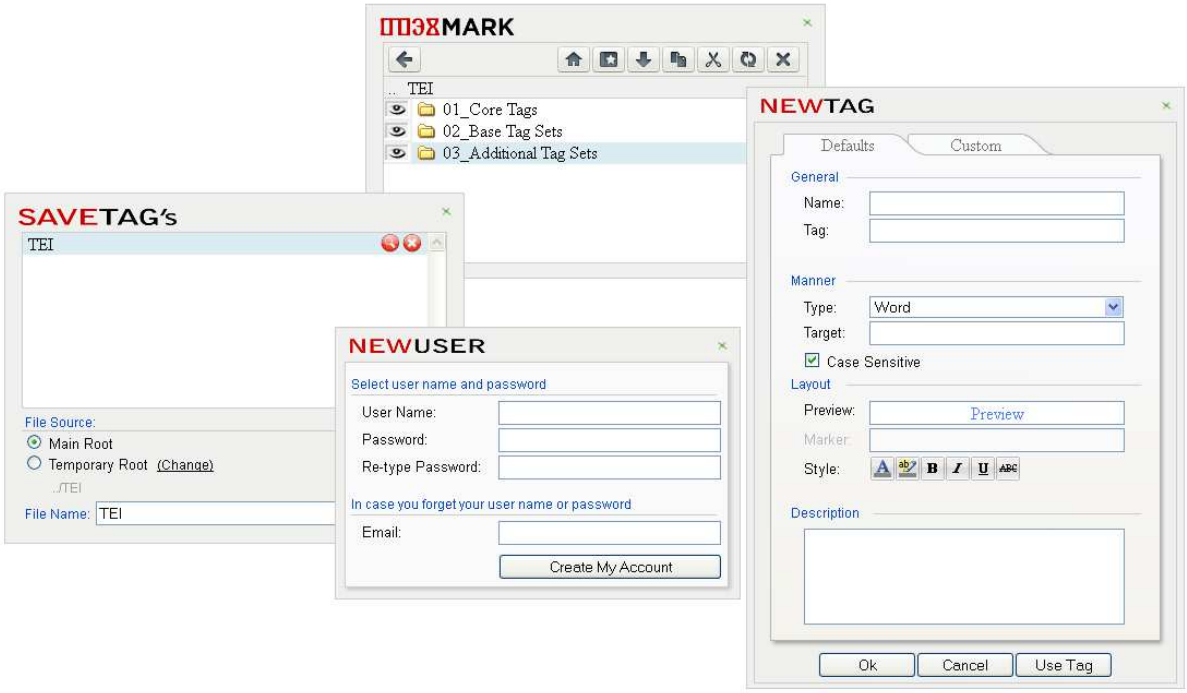
Brisanje TIE elementa se svodi na brisanje njegovih oznaka pomoću temeljnih funkcija. Kad se radi brisanje TIE oznaka ili oznaka pripadajućeg SPAN elementa onda to može biti iz dva razloga. Jedan razlog je kada određeni čvor gubi svojstvo vidljivosti, tada treba pronaći sve njegove TIE elemente unutar teksta i obrisati im pripadajući stil, tj. oznake pripadajućeg SPAN elementa. Drugi slučaj je kada korisnik stvarno želi odznačiti neki označeni pojam, tad je potrebno osim brisanja oznaka određenog SPAN elementa, obrisati i oznake njegovog pripadajućeg TIE elementa.

Moguće dodatne akcije nastaju kad dolazi do brisanja pripadajućeg stila TIE elementa koji se odnosio na pojam koji se križao s nekim drugim označenim pojmom. Rješenje toga problema se sastoji od nekoliko koraka. Prvo je potrebno prepoznati koji čvorovi su se nalazili u tom križanju, zatim treba ukloniti oznake SPAN elementa koji predstavlja to križanje i nakon toga je dovoljno zadati gašenje i paljenje (osvježavanje) stila onim čvorovima koji su se nalazili unutar tog križanja. Nakon gašenja i ponovnog uključivanja stila, odgovarajuće SPAN oznake će se same podesiti na svoje pozicije preko već izgrađenih funkcija za umetanje TIE elementa, tj. njegovog pripadajućeg stila.

6. Mogućnosti TEIMark aplikacije

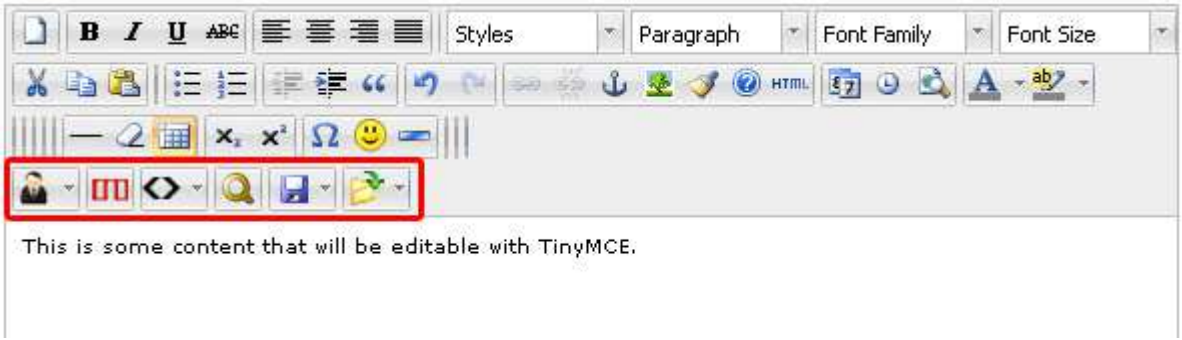
6.1. TEIMark korisničko sučelje

Kao i mnoge današnje aplikacije, tako i TEIMark komunicira sa korisnikom pomoću izgrađenog grafičkog korisničkog sučelja koje se sastoji od raznih dijaloških okvira (kartica) koje olakšavaju korisniku zadavanje željenih radnji.



Slika 9: TEIMark pop-up prozori

Osnovno TEIMark korisničko sučelje ugrađeno je u donjoj alatnoj traci tinyMCE korisničkog sučelja, gdje se nalaze osnovni gumbi koji su poveznica s dijaloškim okvirima.

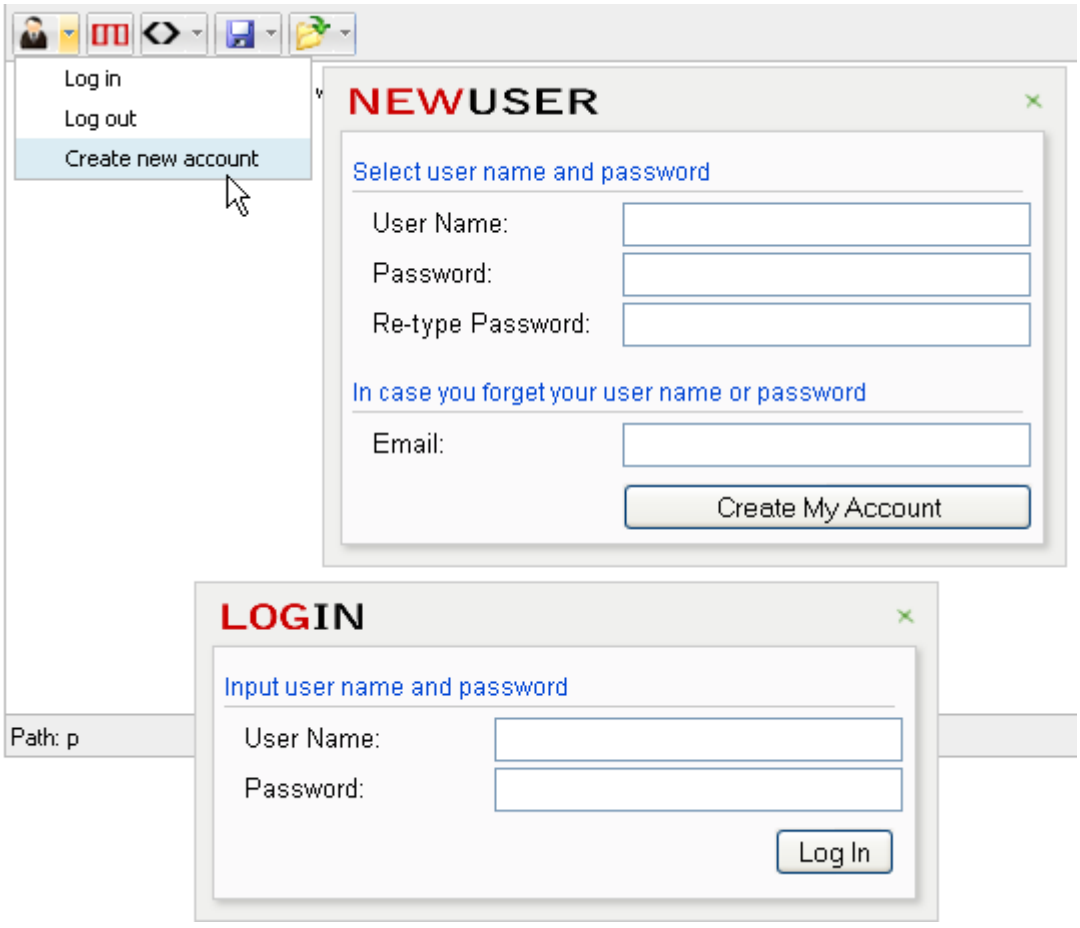


Slika 10: Osnovno TEIMark korisničko sučelje

6.2. Korisnički profili

TEIMark omogućuje korisnicima napraviti vlastiti korisnički profil na kojem će moći spremati označene dokumente i oznake koje su definirali. Otvaranje vlastitog korisničkog profila nije nužno da bi se TEIMark koristio jer je mogućua i pohrana podataka direktno u memoriju korisničkog računala. Prednosti spremanja podataka na vlastito računalo je u tome što je iste moguće slati drugim korisnicima, koji ih onda mogu koristiti u vlastite svrhe. S druge strane, prednost korisničkog profila i spremanja podataka na WEB je zbog mogućnosti pristupa vlastitim podacima gdje god postoji pristup internetu.

Kartice za otvaranje novog korisničkog profila i kasniji pristup istom nalaze se u meniju prvog gumba.

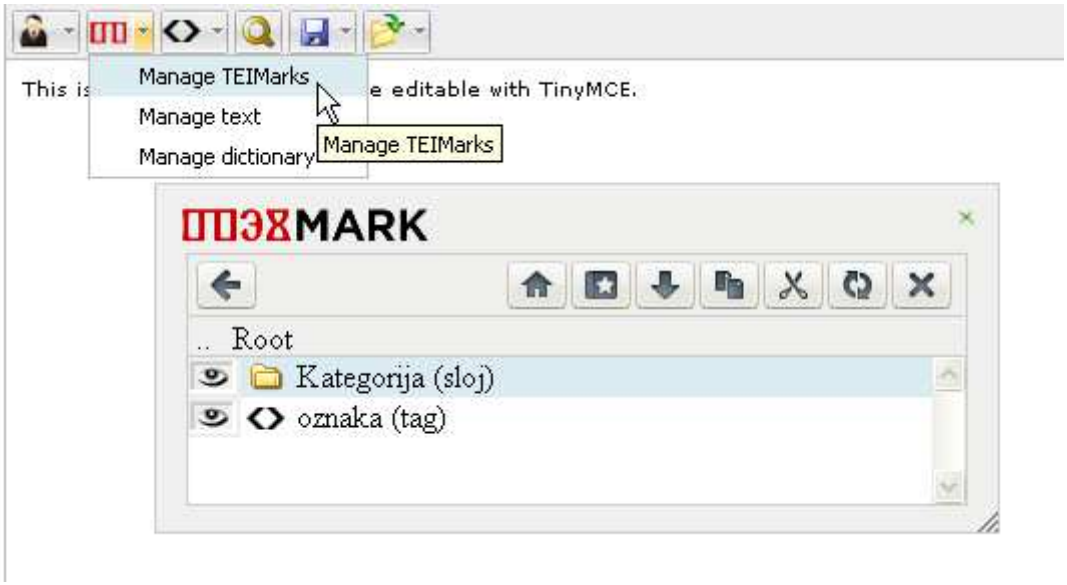


Slika 11: Sučelje za upravljanje korisničkim profilima

Sve što je potrebno za otvaranje novog korisničkog profila je odabrati ime za pristup računu i lozinku. Davanje e-maila je preporučljiva mogućnost, jer ako korisnik zaboravi lozinku, onda može poslati zahtjev za novom lozinkom koju će primiti na tu adresu.

6.3. Upravljanje slojevima

Kartica „TEI MARK“ je sučelje za upravljanje slojevima, a nalazi se u izborniku drugog gumba pod nazivom „Manage TEIMarks“.



Slika 12: Kartica za upravljanje oznakama

Unutar ovog sučelja korisniku je omogućeno :

- stvaranje novih slojeva (kategorija)
- navigacija kroz slojeve
- stvaranje novih oznaka (tagova).
- kopiranje, premještanje i brisanje slojeva ili oznaka
- upravljanje vidljivošću slojeva

Vidljivost stila označenog teksta upravljamo preko simbola "oka" (lijevo od sloja ili oznake):

- ugašeno oko – ne prikazuje se stil označenog objekta (dio riječi, riječ ili skupina slova)
- upaljeno – prikazuju se

Slojevi su hijerarhijski složeni do bilo koje dubine, a unutar bilo kojeg sloja mogu se spremati oznake i drugi slojevi. Uključenje ‘oka’ odnosi se na taj sloj i sve njegove podslojeve.

Način upravljanja slojevima i oznakama sličan je radu u Windows explorer-u. Svaki gumb služi za neku akciju - dolaskom kazala miša iznad ikone ispisuje se pridružena akcija, a klikom na gumb/ikonu akcija se izvodi:

Slika 13: Alatna traka za organizaciju slojeva

Slijedi kratak opis pojedinačnih gumbova:

 - povratak u prethodni sloj. Ulazak u sloj je dvostruki klik na ime sloja.

 - stvaranje novog sloja koji će sadržavati oznake ili nove slojeve.

 - stvaranje nove oznake, otvaranje kartice 'New Tag'.

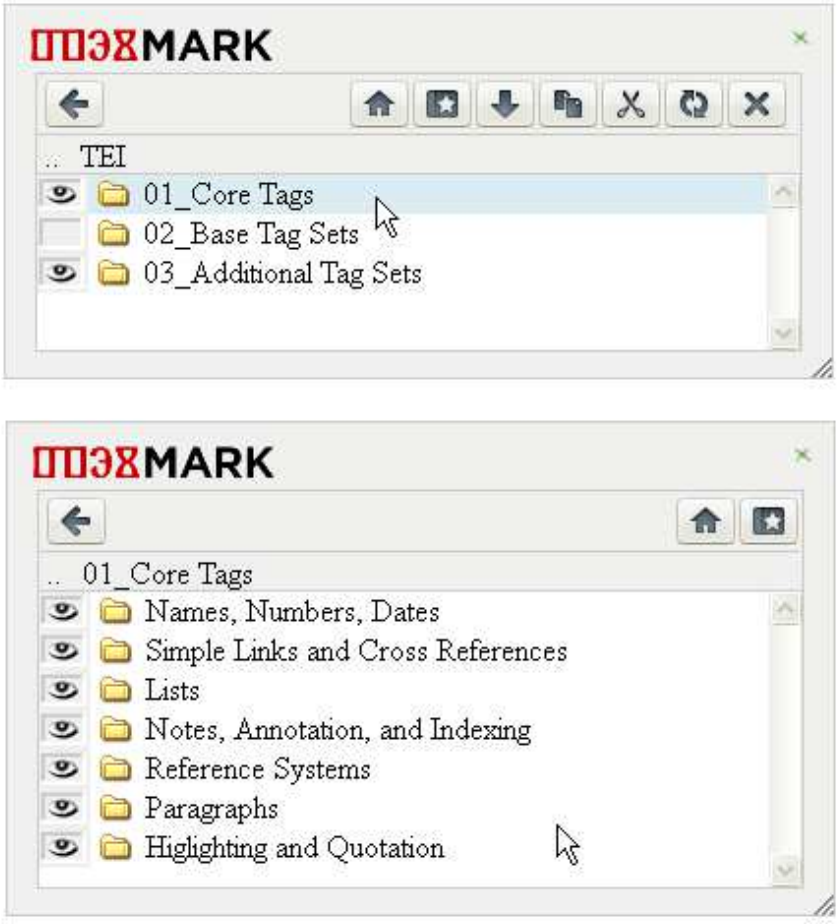
 - umetanje/lijepljenje privremeno spremljene oznake ili sloja (sa svim podslojevima) na novo mjesto.

 - pravljenje kopije neke oznake ili sloja.

 - rezanje/brisanje neke oznake ili sloja iz popisa, uz njeno istodobno spremanje . u privremeno spremište.

 - promjene sloja ili oznake, editiranje.

 - brisanje odabrenog sloja ili oznake. U slučaju brisanja sloja koji ima podstrukturu ili više oznaka u sebi, program će upozoriti želimo li to doista učiniti.

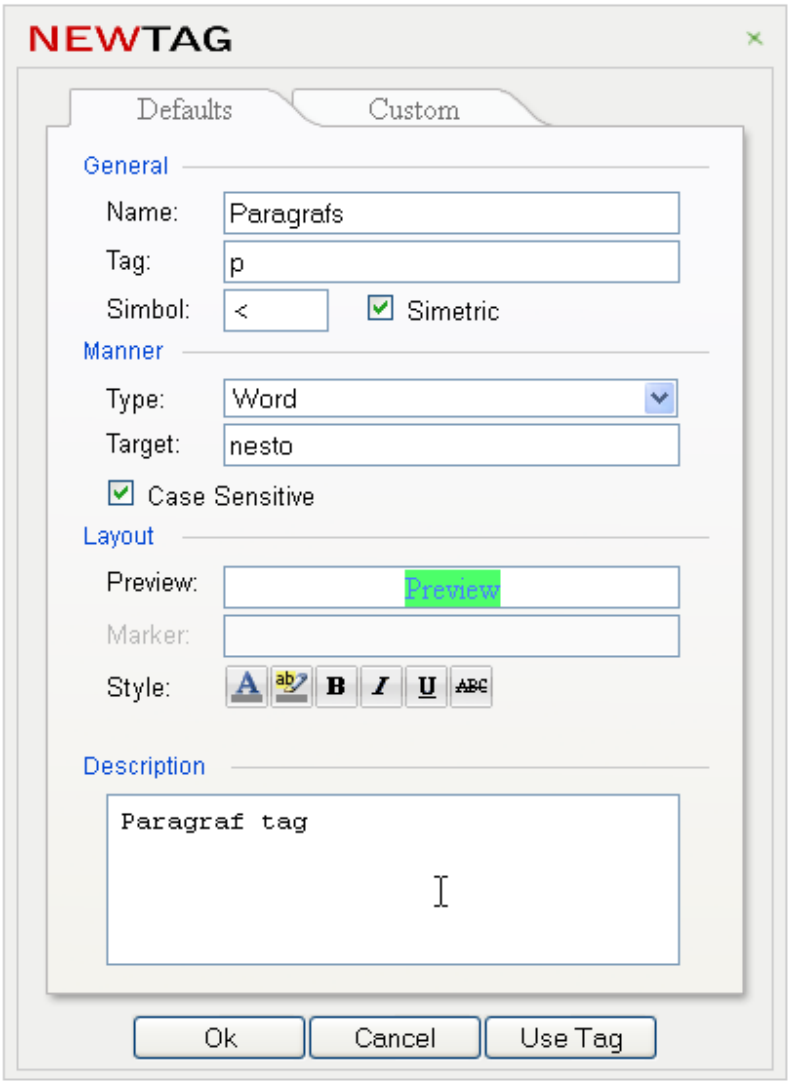


Slika 14:Primjer TEImark kartice i hijerhije slojeva/kategorija

Za stvaranje novog sloja potrebno je definirati njegovo ime:

Slika 15: Kreiranje novog sloja

Za stvaranje nove oznake moguće jedefinirati mnoga svojstva oznake putem kartice „New Tag“:



Slika 16: Kartica za opis novog taga

Unutar kartice „New Tag“ nalaze se dvije podkartice:

- Default-za definiranje osnovnih svojstava svakog elementa oznake
- Custom-za definiranje argumenata oznake

U prvoj podkartici „Default“, svojstva koja se mogu definirati su:

General

- **Name** - ime koje se odnosi na stvorenu oznaku u trenutačno otvorenom sloju u kojem će oznaka biti pohranjena
- **Tag –** ime elementa koji će se prilikom izvoza (exporta) teksta prikazat unutar zagrada "<>" , ili nekog drugog definiranog simbola.
- **Simbol –** skup znakova koji ograđuje samo ime elementa.
- **Symetric** - aktivator simetrije simbola oznake, ako ista posjeduje svoju simetriju.

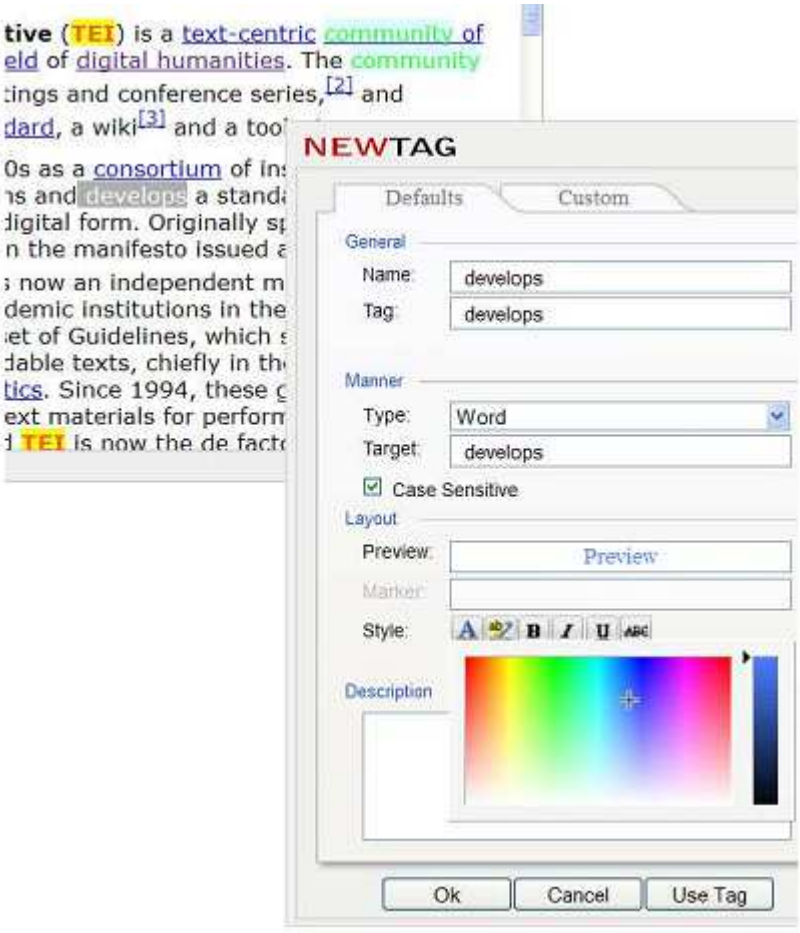
Manner

- **Type –** definiranje vrste oznaka u ovisnosti o namjeni. Postoji 6 različitih namjena oznaka, koje su podjeljene u dvije glavne kategorije: „vezane“ i „slobodne“ oznake. Vezane oznake okarakterizirane su sa svojstvom „target“, tj. moguće ih je koristiti samo za označivanje unaprijed predefiniranih objekata (meta). To svojstvo ima veliku prednost kod automatskog označivanja tekstova. Moguće vrste vezanih oznaka:
  1. Word - obuhvaća (wrap) slova, riječi ili tekst, klasično označavanje.
  2. Point - umeće oznaku na neko mjesto (npr. između dva slova), umjesto ‘uokolo’ znakova. To može biti npr. umetanje argumenta oznake na mjesto unutar skraćenice (poput ‘o’ atributa oznake izmedju slova ‘B’ i ‘g’ koja nedostaje riječi ‘Bog’) ili umetanja nove riječi, novog retka, paragrafa i slično.
  3. Phrase - određenih skupina riječi ili znakova u tekstu koje mogu sadržavati umetnute nepoznate riječi poput: "Ja hodam", "Ja brzo hodam", "Ja ponekad sporo hodam". U sva tri slučaja program bi mogao prepoznati frazu "Ja hodam" i na njoj primjeniti zadanu oznaku (graf – povezivanje nesusjednih riječi ili njihovih dijelova)

  Moguće vrste slobodnih oznaka:
  1. Free Word
  2. Free Point
  3. Free Phrase
- **Target** - skup simbola (znakova ili riječi) koje program automatski pronalazi u tekstu ovisno o tipu, tj. namjeni korištene oznake.
- **Case Sensitive** - osjetljivost oznake na velika-mala slova.

Layout

- **Preview** - prikazuje kako će izgledati stil označenih riječi tijekom rada. Stilom se npr. definira boja slova objekta (targeta) i njegove pozadine. U označenom tekstu ne vide se simboli oznaka (<>...</>) nego su objekti označeni pripadnim stilom, što je puno ugodnije za rad. Riješeno je i preklapanje oznaka koje XML ne dopušta (well-formedness), a TEIMark rješava. U slučaju isključenja 'oka' u TEIMark kartici, sve oznake iz tog sloja se ne prikazuju u tekstu niti izvoze u XML-zapisu. Tako je riješen vrlo teški problem kolaboracije autora nad istim tekstom u svim varijantama (isti tekst – isti sloj, isti tekst-različiti slojevi).
- **Marker** - dostupan je samo za oznake vrste POINT. Marker je skup simbola koji će s prikazivati na poziciji oznake vrste POINT.
- **Style** - sučelje za upravljanje stilom označenih riječi. Na raspolaganju su: boja slova, pozadinska boja, podebljana, nakošena, podcrtana i prekrižena slova. Njihovom kombinacijom dobivamo mogućnost dobrog razlikovanja pridruženog označivanja na zadanom tekstu.

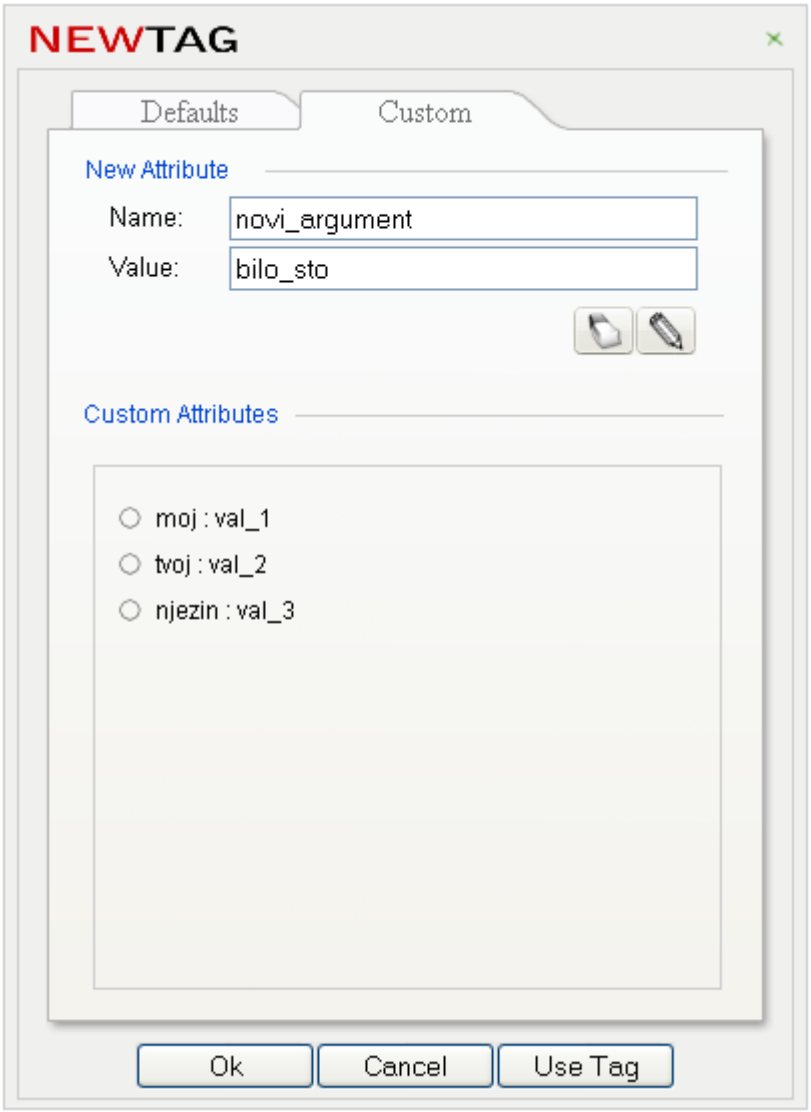


Slika 17: Prikaz stila označenog teksta

- **Description** - slobodni opis oznake koji će biti prikazan u info kartici pojedine oznake, skupa sa svim ostalim svojstvima.

“Custom” podkartica služi za stvaranje vlastitih atributa za željenu oznaku. Svakom atributu dodjeljujemo njegovo ime (NAME) i vrijednost (VALUE).

Parove ‘name:value’ možemo brisati (ikona s gumicom) ili spremati (ikona s olovkom). U slučaju spremanja par će se naći u popisu ‘Custom attributes’. Ti će se atributi naći zajedno s imenom elementa (oznake) kod exportiranja označenog teksta. <ime_elementa arg1=val1 , arg2=val2, ..> označeni tekst </ime_elementa>

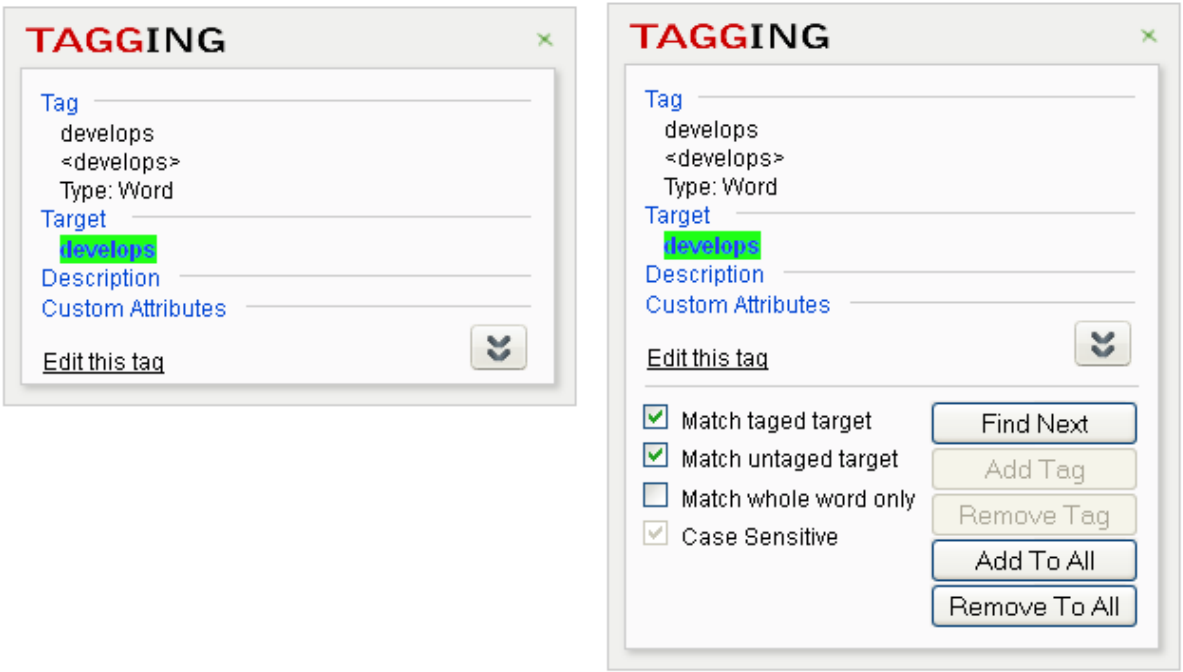


Slika 18: Podkartica "Custom"

Nakon što je završeno definiranje oznake, ona se sprema klikom gumba „Ok“ u aktivan sloj, tj. onaj sloj koji je trenutačno otvoren na kartici ‘TEI MARK’. Moguće je također direktno koristiti oznake uz automatsko spremanje klikom gumba "Use Tag".

6.4. Označivanje teksta

Za označivanje teksta koristimo oznake koje smo unaprijed definirali kao što je opisano u prethodnom poglavlju. Za označivanje je potrebna kartica „TAGGING“, koja se otvara dvostrukim klikom na željenu oznaku unutar kartice „TEI MARK“ ili klikom na gumb „Use Tag“ unutar kartice za stvaranje nove oznake.



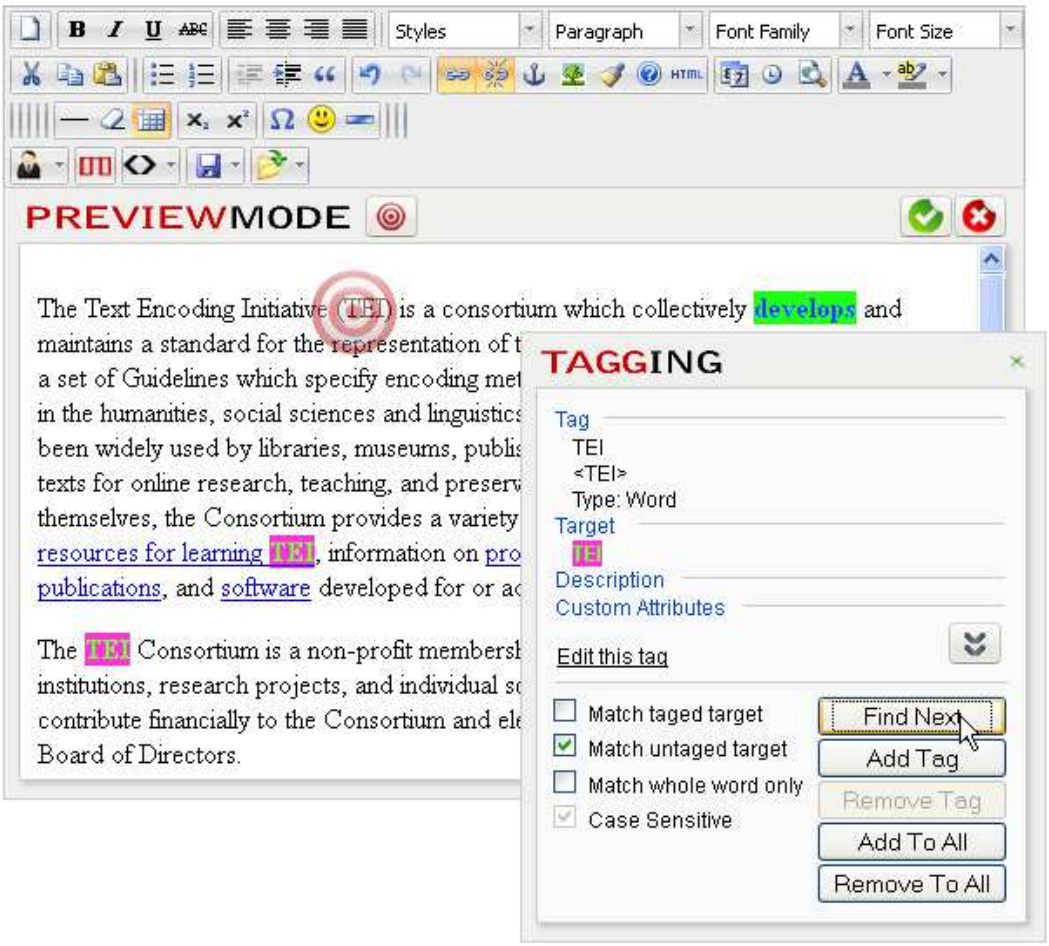
Slika 19: Primjer kartice "TAGGING" za oznake vezanog tipa

Kartica „TAGGING“ može biti zatvorena ili otvorena. Ako je zatvorena, onda se prikazuje samo gornji dio kartice čija je glavna namjena da pokaže osnovne informacije o korištenoj oznaci. U donjem lijevom kutu se nalazi opcija „Edit this tag“, koja omogućuje da se korištena oznaka mijenja (editira) i putem ove kartice. U donjem desnom kutu se nalazi gumb koji proširuje karticu „TAGGING“ i nudi korisniku dostupne opcije za označivanje teksta. Taj dio kartice se razlikuje kod oznaka koje su vezanog ili slobodnog tipa.

Označivanje teksta s vezanim oznakama

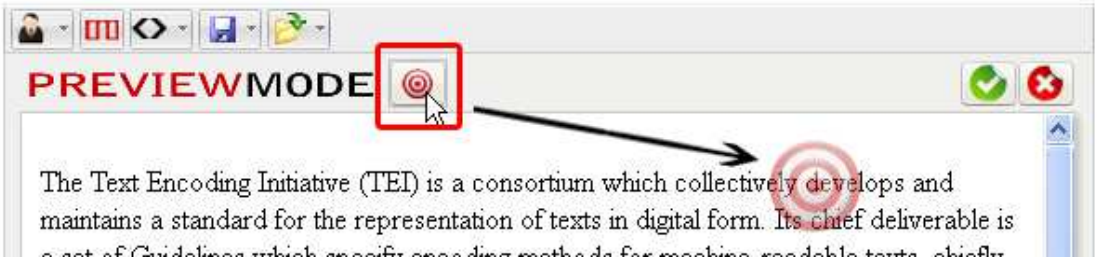
Ukoliko se radi o označivanju s oznakama vezanog tipa, „TAGGING“ kartica izgleda kao na slici 18, a značenje ponuđenih funkcija je:

- **Find Next** - pronalazi prvi sljedeći traženi objekt u tekstu. Ako se radi o prvoj pretrazi, onda se tekst editor prebacuje u „Preview Mode“.



Slika 20: Preview mode

- **Add Tag** – označuje pronađeni objekt. Ova opcija je dostupna samo nakon pretrage „Find Next“, i to ukoliko je pronađen željeni target. Ako nismo sigurni gdje se nalazi pronađeni objekt, možemo ga locirati prtiskom na ikonu mete, u alatnoj traci Preview moda.



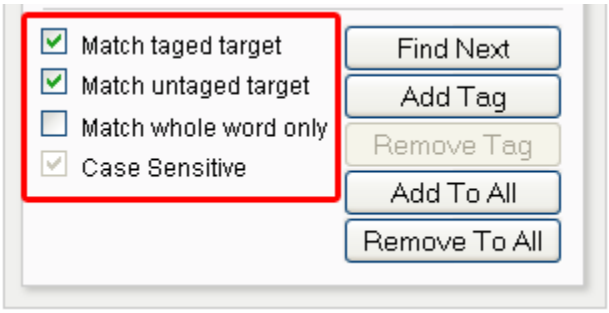
Slika 21: Previw mode lokator

- **Remove Tag** – uklanja oznaku pronađenom objektu. Ova opcije je dostupna samo nakon pretrage, ako je pronađeni objekt već označen.
- **Add To All** – pronalazi i označuje sve objekte u tekstu.
- **Remove To All** – pronalazi sve objekte i uklanja oznaku ako su označeni.

Nakon završetka označivanja povratak u tekst editor omogućuje se pomoću gumbića u alatnoj traci Preview moda , gdje se mogu spremiti ili odbaciti sve načinjene promjene.

Slika 22: Gumbi za prihvaćanje ili odbacivanje promjena označivanja

Osim prethodno objašnjenih funkcija, kartica „TAGGING“ sadrži pojedine aktivatore o kojima ovisi pretraga pozvana klikom na „Find Next“.



Slika 23: Aktivatori o kojima ovisi pretraga

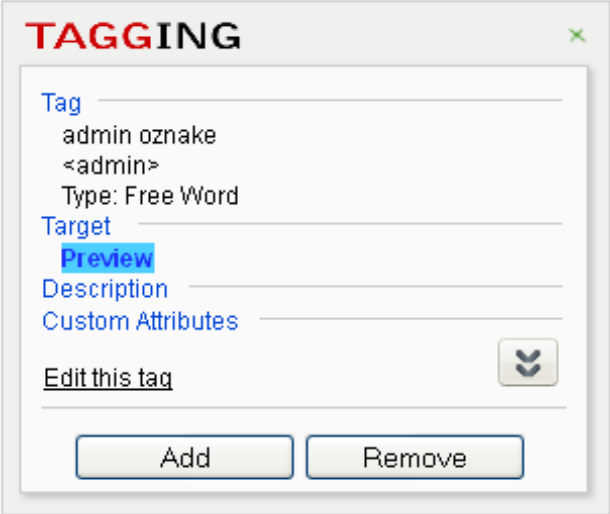
Postoje tri aktivatora:

- **Match taged target** – ako je uključen, pretraga će pronalaziti objekte koji su već označeni.
- **Match untaged target** - ako je uključen, pretraga će pronalaziti objekte koji nisu označeni.
- **Match whole word only** – ako je uključen pretraga će pronalaziti samo one objekte koji čine kompletnu riječ.

Case Sensitive – je indikator osjetljivosti oznake na velika i mala slova. Njegova vrijednost je dobivena na temelju odabira kod početnog stvaranja oznake i u ovoj kartici njegova promjena nije moguća.

Označivanje teksta sa slobodnim oznakama

Označivanje teksta može se izvesti i pomoću slobodnih oznaka. U tom slučaju „TAGGING“ kartica izgleda kao na sljedećoj slici:



Slika 24:Primjer kartice „TAGGING“ za oznake slobodnog tipa

Kod slobodnih oznaka, označivanje teksta i uklanjanje oznaka se izvodi samo pomoću dva gumba. Za umetanje oznaka je dovoljno označiti (selektirati) dio teksta kojeg želimo označiti, te nakon toga kliknuti gumb „Add“. Za brisanje oznaka je dovoljno selektirati dio teksta u kojem se nalazi jedna ili više oznaka , te nakon toga kliknuti gumb „Remove“.

Kao što je već spomenuto, slobodne oznake mogu označivati bilo koji dio teksta, ali jednom označeni dio teksta se ne smije više mijenjati kako bi zadržao pripadajuću oznaku. Ukoliko se označeni dio teksta svejedno promjeni, pripadajuća oznaka će se kasnije sama obrisati prilikom spremanja dokumenta ili poziva funkcija koje izvršavaju određene operacije nad označenim tekstom.

Pregled korištenih oznaka

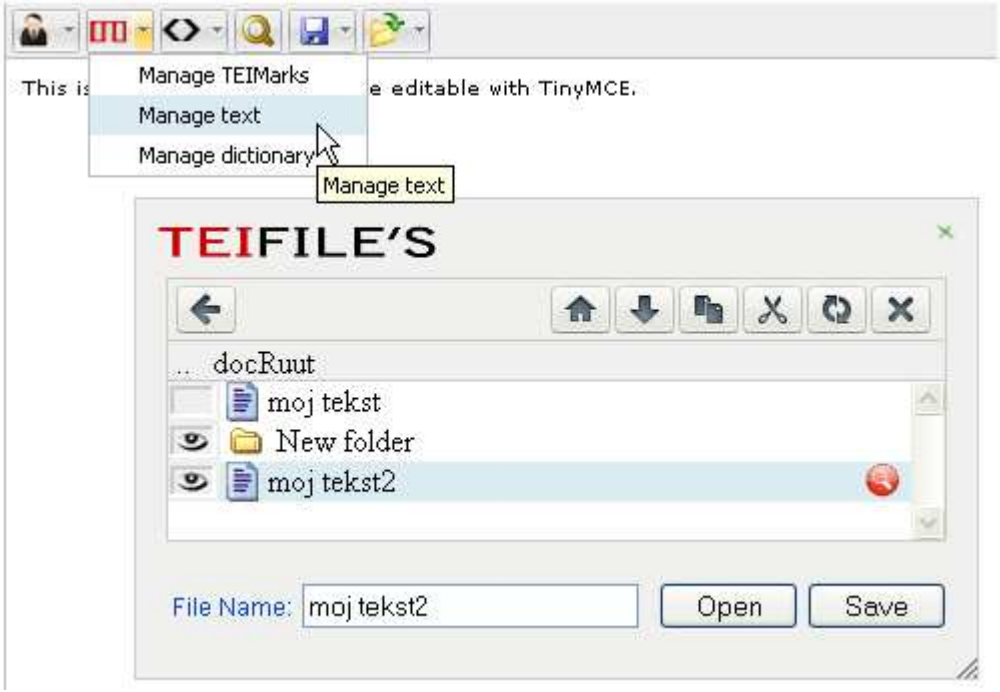
Unutar Preview moda je također moguće pogledati informacije o oznakama koje smo koristili prilikom označivanja teksta. Za otvaranje „TAGGING“ kartice neke oznake koju smo već koristili, dovoljno je kliknuti na riječ nad kojom je ta oznaka primjenjena. Ukoliko se klikne na riječ koja je označena sa više različitih oznaka , otvorit će se kartica „SUGGESTED TAGS“ koja prikazuje sve oznake s kojima je ta riječ označena. Prelaskom kazala miša preko naziva oznake unutar kartice „SUGGESTED TAGS“, sa strane se otvara pripadajuća „TAGGING“ kartica.

Također je moguće i putem „SUGGESTED TAGS“ kartice upravljati vidljivošću stila označenog teksta, klikom na malu ikonu „oka“ lijevo od naziva oznake.

Slika 25: Kartica "SUGGESTED TAGS"

6.5. Pohrana označenih tekstova

Označeni tekst je moguće spremiti unutar programa u izvornom formatu kako bi se omogućio kasniji nastavak označivanja istog teksta. Kartica za upravljanje označenim tekstovima se nalazi u izborniku drugog gumba pod nazivom „Manage text“.



Slika 26: Kartica za upravljanje tekstovima

Kartica za upravljanje tekstovima omogućava strukturalnu pohranu tekstova unutar foldera, kopiranje, rezanje i brisanje podataka. Ali međuostalim opcijama je zadržano i svojstvo viljivosti podataka (ikona oka), koje u ovom slučaju određuje hoće li vidljivi podaci biti uključeni u određene aktivnosti koje će zahtjevati korisnik.

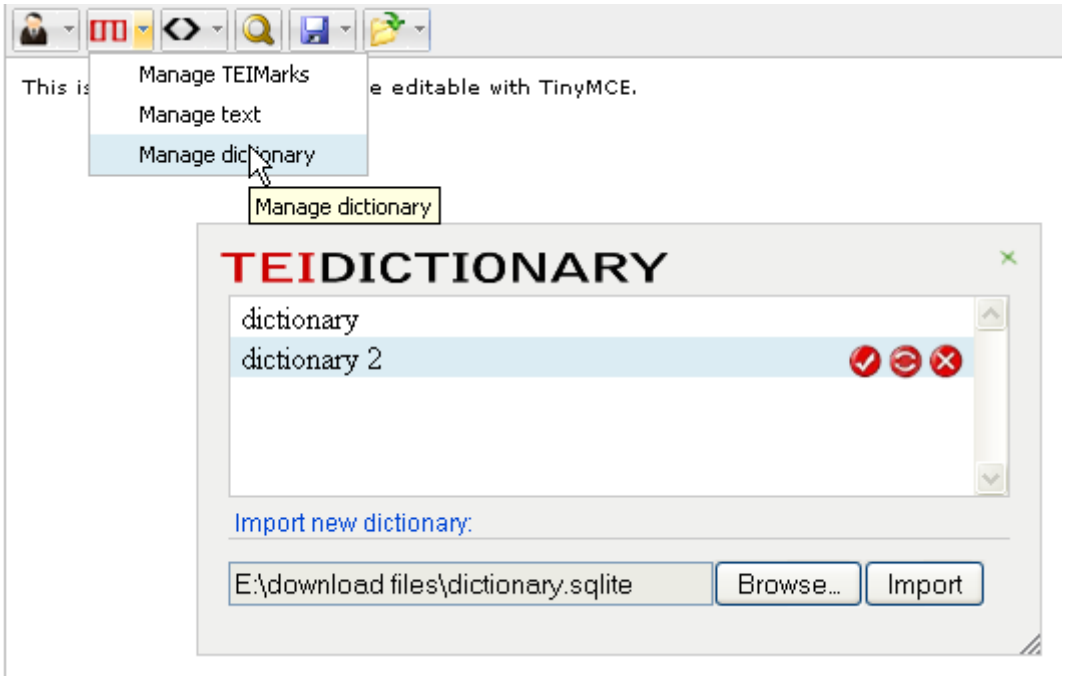
Za spremanje aktualnog teksta unutar editora, dovoljno je upisati željeni naziv podatka i kliknuti gumb „Save“. Ukoliko upišemo već postojeće ime, program će nam nuditi opciju presnimavanja istoimenog podatka.

Prije otvaranja željenog tekstualnog podatka, isti se može pogledati pomoću male ikone povećala desno od naziva podatka.

6.6. Zamjena riječi pomoću baze podataka-rječnika

Zamjena riječi pomoću baze podataka omogućava zamjenu svake riječi u tekstu s onim pojmom koji je asociran uz tu riječ preko odgovarajućeg riječnika. Korisniku je omogućeno da postavlja na svoj korisnički račun vlastite baze podataka prema kojima želi izvršavati zamjenu riječi.

Sučelje koje omogućava učitavanje rječnika i obradu teksta prema istima se nalazi u izborniku drugog gumba pod nazivom „Manage dictionary“.



Slika 27: Kartica za upravljanje rječnicima

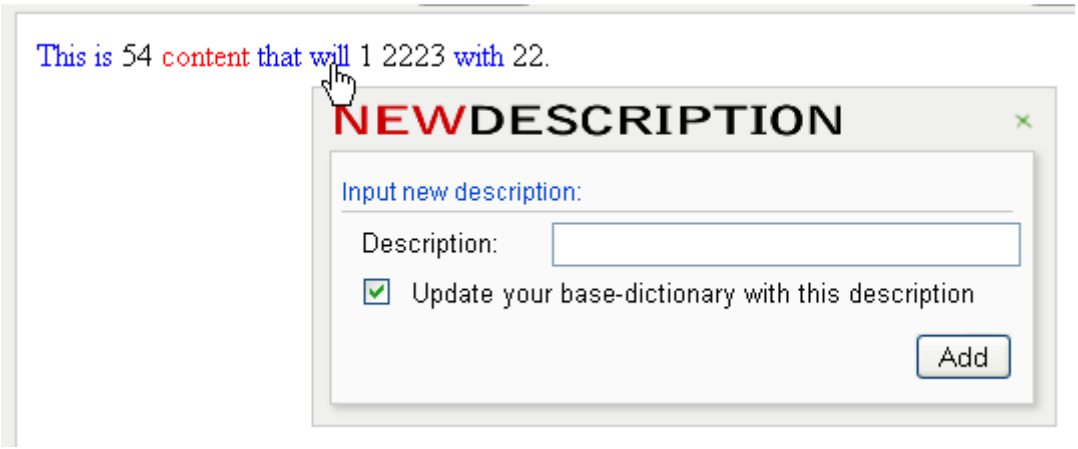
Uvjeti koje baza podataka mora ispunjavati kako bi bila kompatibilna za sustav je da se nalazi u „sqlite“ formatu, naziv tablice koja sadrži riječi i pripadajuće opise mora biti „dictionary“, a kolone u kojima se nalaze riječi i pripadajući opisi se moraju nazivati „word“ i „description“.

Dvostrukim klikom na određeni rječnik ili klikom na ikonu kvačice desno od imena rječnika, započinje zamjena rječi. Nakon što server obradi podatke, editor se prebacuje u „Preview mode“ gdje se vide nastale promjene nad tekstom nakon prve faze zamjene riječi.

Slika 28: Prva faze zamjene riječi pomoću rječnika

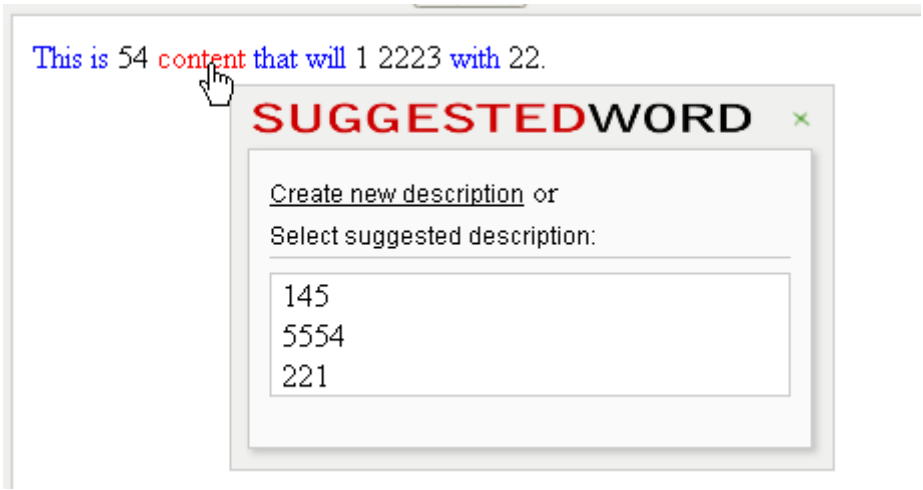
Nakon prve faze zamjene riječi, program je zamjenio samo one riječi koje su jednoznačno definirane u rječniku. One riječi za koje je pronađeno više različitih opisa su markirane crvenom bojom, dok su plavom bojom markirane riječi za koje nije pronađen niti jedan opis.

Klikom miša na plave riječi otvara se kartica „new description“ koja nudi upis novog opisa te riječi, sa mogućnošću da se novi opis te riječi ažurira u bazu.



Slika 29: Kartica za unos novog opisa riječi

Klikom miša na crvene riječi otvara se kartica koja nudi izbor između već postojećih opisa te riječi ili otvaranje kartice „new description“, za unos novog opisa.



Slika 30: Kartica "suggested word"

Kako bi se korisnik lakše snalazio unutar teksta sa zamjenjenim riječima, omogućen mu je prelazak iz originalnog teksta u obrađeni tekst i obratno, pomoću gumba „switch mode“ koji se nalazi u alatnoj traci preglednika.

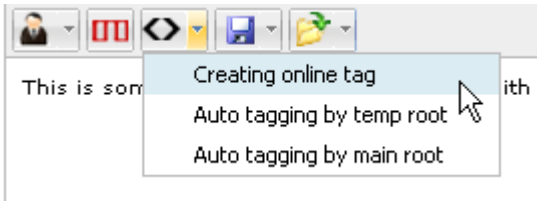
Slika 31: Alatna traka unutar „preview mode“

Nakon prebacivanja teksta u orginalni oblik i dalje je zadržana mogućnost upisivanja novog opisa riječima ili izbora opisa kod riječi sa više opisa.

Nakon što je korisnik zadovoljan sa postignutom zamjenom riječi, ima mogućnost spremanja novog teksta na vlastito računalo pomoću ikone diskete koja se nalazi u istoj alatnoj traci.

6.7. Brzo stvaranje oznaka

Brzo stvaranje oznaka je mogućnost koja pojednostavljuje unošenje osnovnih svojstava oznaka kod stvaranja nove oznake vezanog tipa. Također opcija za brzo kreiranje oznaka je postavljena tako da je pristupačnija korisniku u odnosu na klasično stvaranje nove oznake, a ujedno korisnik prilikom stvaranja oznake dobiva i povratnu informaciju dali slučajno već postoji slična oznaka onoj kakvu želi napraviti. Brzo stvaranje oznaka se nalazi u meniju trećeg gumba osnovnog TEIMark korisničkog sučelja pod opcijom „Creating online tag“.



Slika 32: Opcija brzog kreiranja oznaka

Za brzo stvaranje oznaka je prvo potrebno selektirati dio teksta kojeg želimo postaviti za objekt (target) oznake koju namjeravamo napraviti , a zatim treba kliknuti opciju „Creating online tag“ što će rezultirati otvaranjem kartice „New Tag“ za stvaranje nove oznake s predefiniranim osnovnim svojstvima sukladno selektiranom tekstu.

Nakon što se pozove funkcija „Creating online tag“, program provjerava već postojeću bazu oznaka, te ukoliko postoje oznake koje dijele zajednički objekt (target) sa selektiranim tekstom, otvorit će se kartica „SUGGESTED TAGS“ s prikazom pronađenih oznaka i sa mogućnošću odabira stvaranja nove oznake.

Ukoliko je pozvana funkcija „Creating online tag“, a prethodno nije selektiran niti jedan dio teksta, program pretpostavlja da korisnik želi stvoriti oznaku vrste „Point“, te prema poziciji „kursora“ unutar teksta ovise predefinirana svojstva oznake vrste „Point“.

6.8. Automatsko označivanje teksta

Automatsko označivanje teksta omogućuje označivanje cijelog teksta prema već unaprijed pripremljenoj bazi oznaka, praktički s jednim klikom miša. Oznake koje se koriste za automatsko označivanje teksta moraju biti vezanog tipa, na taj način je omogućeno da svaka oznaka pronalazi svoj objekt u tekstu i označuje ga.

Opcije za automatsko označivanje teksta se nalaze u meniju trećeg gumba osnovnog TEIMark korisničkog sučelja.

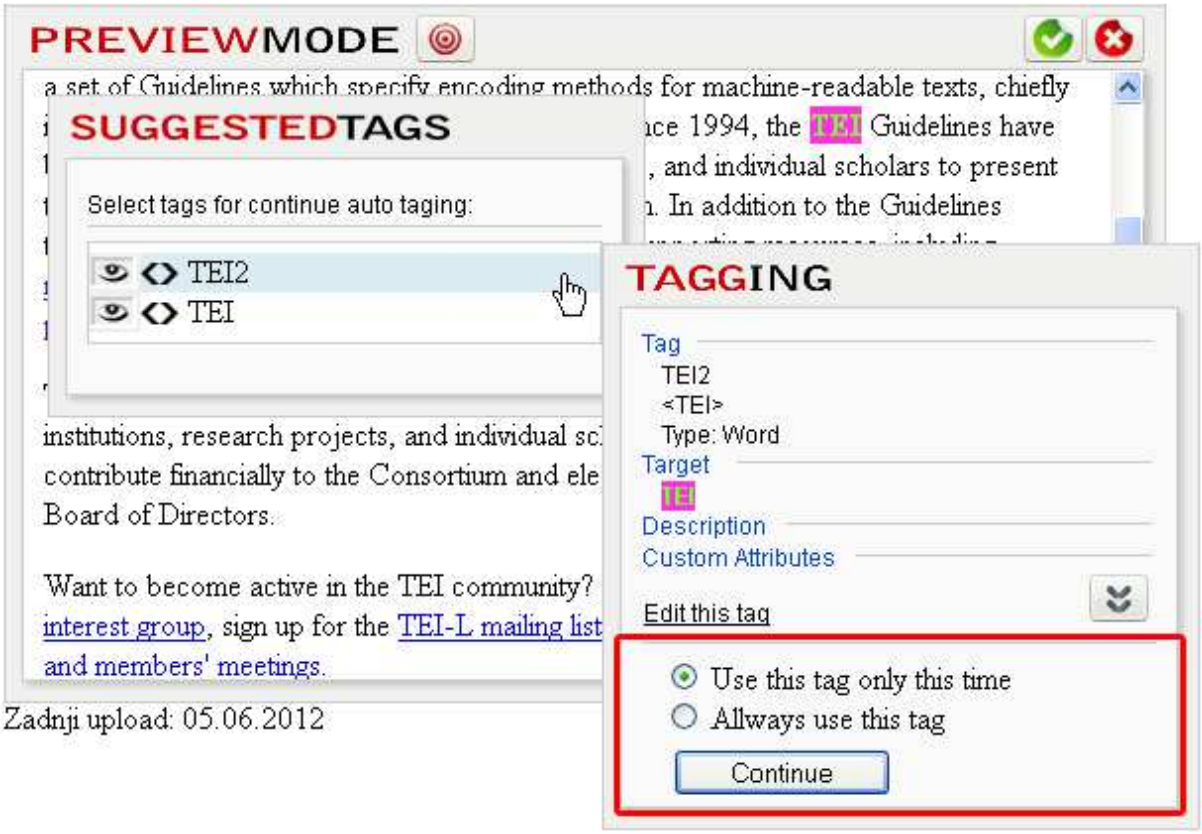
Slika 33: Opcije za automatsko označivanje

Postoje dvije opcije za automatsko označivanje:

- **Auto tagging by temp root** – izvršit će označivanje nad cjelim tekstom koristeći samo one oznake koje se nalaze u aktivnom sloju i njegovim pod slojevima
- **Auto tagging by main root** – izvršit će označivanje nad cjelim tekstom koristeći sve oznake koji se nalaze u kartici „TEIMark“

Obje opcije automatskog označivanja neće koristiti one oznake koje se nalaze unutar ugašenih slojeva.

Kad započne proces automatskog označivanja, tekst editor se prebacuje u „Preview mode“ i nakon završetka procesa potrebno je prihvatiti ili odbaciti sve načinjene promjene. Ukoliko proces automatskog označivanja naiđe na riječ koja odgovara objektima dvaju ili više oznaka, zaustavit će se i pomoću kartice „SUGGESTED TAGS“ će ponuditi korisniku da odabere koja oznaka će se primjeniti na pronađenu riječ. Pomoću kartice „SUGGESTED TAGS“ otvaramo karticu „TAGGING“ koja u ovom slučaju izgleda nešto drugačije, kao što prikazuje sljedeća slika:

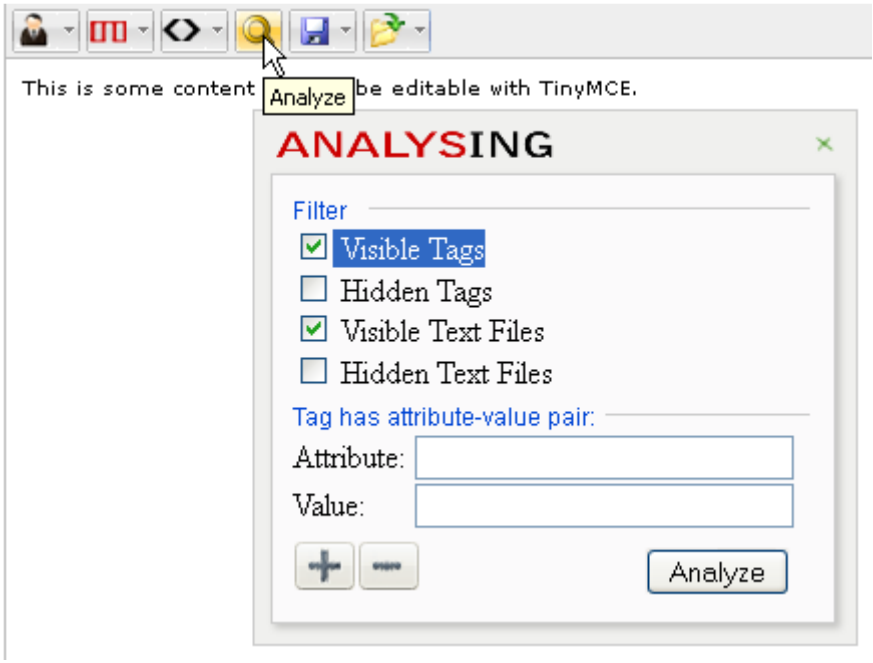


Slika 34:Kartica "TAGGING" kod automatskog označivanja

Kod automatskog označivanja, kartica „TAGGING“ ima opciju „Continue“ s kojom nastavljamo automatsko označivanje s onom oznakom na čijoj kartici smo odabrali nastavak procesa. U slučaju da se proces automatskog označivanja često zaustavlja na istoj riječi koja odgovara objektima više različitih oznaka, a korisnik želi uvijek primjeniti istu oznaku za označivanje te riječi, postoji mogućnost da isključi daljnje zaustavljanje procesa na toj riječi pomoću aktivatora „Allways use this tag“ koji se također nalazi na „TAGGING“ kartici.

6.9. Prebrojavanje oznaka

Pomoću opcije za prebrojavanje oznaka, korisnik ima mogućnost uvida u sve oznake koje je koristio za označivanje tekstova koji se nalaze na njegovom korisničkom računu. Kartica za prebrojavanje oznaka se otvara klikom na četvrti gumb osnovnog TEIMark korisničkog sučelja.



Slika 35: Kartica za prebrojavanje oznaka

Unutar kartice za prebrojavanje oznaka postoje četiri aktivatora pomoću kojih se može prilagoditi pretraga oznaka prema želji korisnika:

- Visible Tags –program uzima u obzir vidljive oznake korištene za označivanje
- Hidden Tags – program uzima u obzir sakrivene oznake korištene za označivanje
- Visible Text Files – program radi pretragu unutar vidljivih tekstualnih podataka
- Hidden Text Files – program radi pretragu unutar sakrivenih tekstualnih podataka

Korisnik također ima mogućnost unošenja parova atributa i njihovih pripadajućih vrijednosti, te na taj način može ograničiti pretragu na samo one oznake koje posjeduju željene atribute.

6.10. Prikaz označenog teksta u željenoj formi

Jednom označeni tekst je moguće izvesti (export) u različite oblike koji su pogodni za daljnju obradu teksta. Trenutno se nude dva različita moguća oblika izvoza koja se mogu naći u meniju četvrtog gumba osnovnog TEIMark korisničkog sučelja.

Slika 36: Opcije eksportiranja teksta

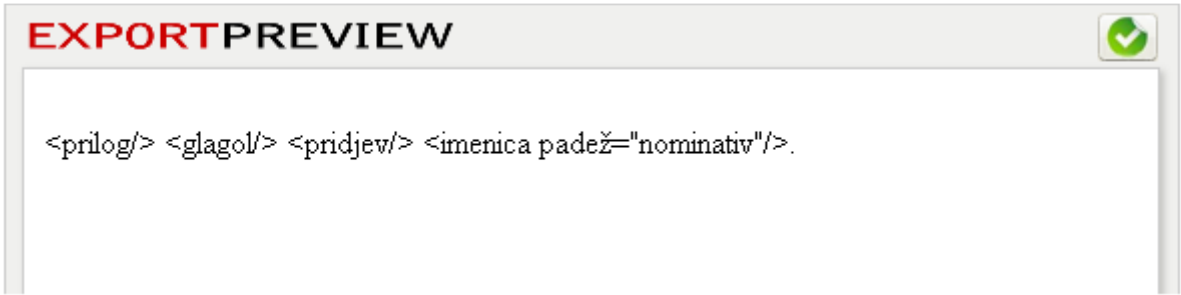
Mogućnosti izvoza teksta su:

- Replaced form
- XML form

Obje opcije uzimaju u obzir samo one oznake koje se nalaze u uključenim slojevima.

Slika 37: Izgled teksta tijekom rada

Prva opcija „Replaced form“ zamjenjuje označene riječi s pripadajućim oznakama i definiranim atributima. Kod takve zamjene je nemoguće zadržati ugnježdenu strukturu oznaka, te ukoliko postoji takva struktura oznaka unutar označenog teksta, ista neće biti izgubljena ali se neće moći prikazati kod ovog izvoza. U slučaju zahtjeva za izvozom teksta koji ima ugnježdenu strukturu, prikazat će se samo one oznake koje su najdublje ugnježdene, a ostale će se ponašati kao da ne postoje.



Slika 38: Prikaz izvora "Replaced form"

Druga opcija „XML form“ pretvara označeni tekst u klasične XML elemente koji za razliku od prve opcije mogu biti međusobno ugnježđeni.

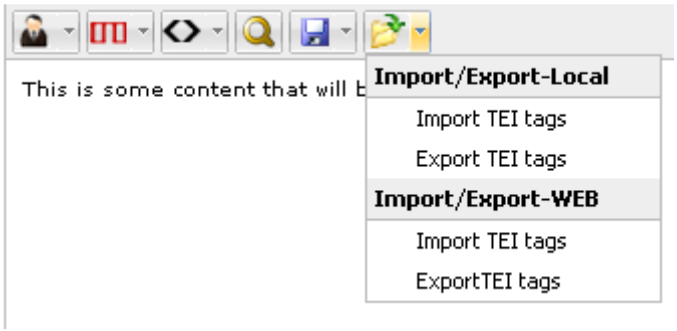
Slika 39:Prikaz izvora "XML form"

Izvoz oznaka koje se križaju

Iako je TEIMark omogućio označivanje teksta s oznakama koje se međusobno križaju, takvu strukturu je naravno nemoguće pretvoriti u ispravan XML i u slučaju križanja oznaka program neće dopustiti izvoz teksta prije nego li se ugase određeni slojevi koji sadrže oznake koje se međusobno križaju.

6.11. Spremanje i učitavanje oznaka

Jednom načinjene slojeve i oznake moguće je spremiti lokalno, tj. na vlastito računalo ili na WEB server, kao i učitati ih s istih. Za spremanje podataka na WEB server je nužno imati otvoren korisnički profil kao što je opisano u poglavlju 6.2. Opcije za spremanje podataka nalaze se u meniju petog gumba osnovnog TEIMark korisničkog sučelja.



Slika 40: Opcije za spremanje slojeva i oznaka

Ponuđene opcije su grupirane u dvije skupine:

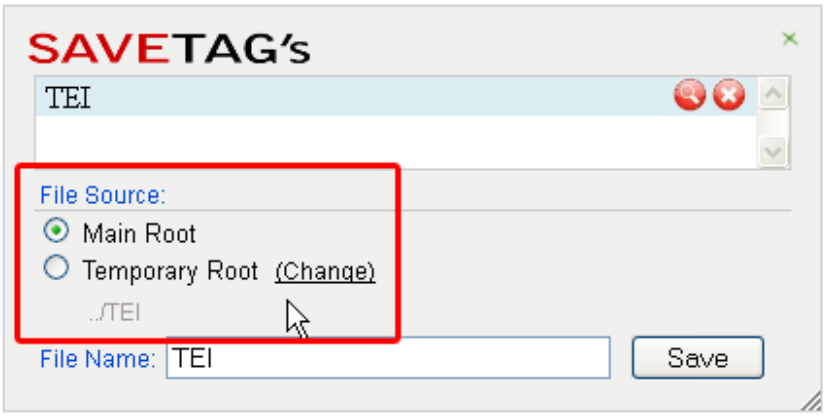
Import/Export-Local

- Import TEI tags – učitavanje oznaka sa vlastitog računala
- Export TEI tags – spremanje oznaka na vlastito računalo

Import/Export-WEB

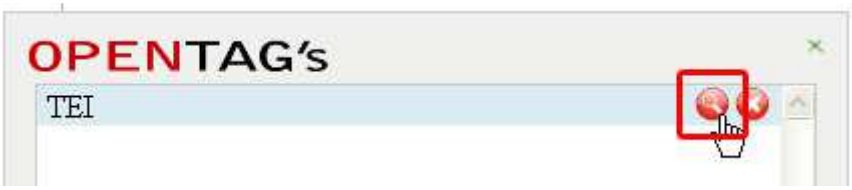
- Import TEI tags – učitavanje oznaka sa WEB servera
- Export TEI tags – spremanje oznaka na WEB server

Bez obzira radilo se o lokalnom ili WEB spremanju podataka, korisnik ima mogućnost spremiti sve oznake i slojeve koji se nalaze unutar „TEIMark“ kartice ili samo one oznake i slojeve koji se nalaze unutar aktivnog sloja . Isto kao i kod učitavanja oznaka, korisnik može birati hoće li oznake biti učitane u glavni korijen svih slojeva (main root) ili će biti učitane u trenutno aktivan sloj (temporary root).



Slika 41: Kartica za spremanje oznaka

Prije učitavanja oznaka s WEB servera, korisniku je omogućeno detaljno pogledati koje oznake planira učitati i prije nego ih stvarno učita u „TEI MARK“ karticu. Ta opcija se poziva klikom na malu ikonu povećala desno od naziva podatka koji se učitava.



Slika 42: Opcija za prikaz oznaka sa WEB servera

Nakon klika na ikonu povećala otvara se kartica „PREVIEW TAG“ koja je izgledom gotovo identična kao „TEIMark“ kartica, ali služi samo za navigaciju kroz oznake.

Slika 43: Kartica "PREVIEW TAG"

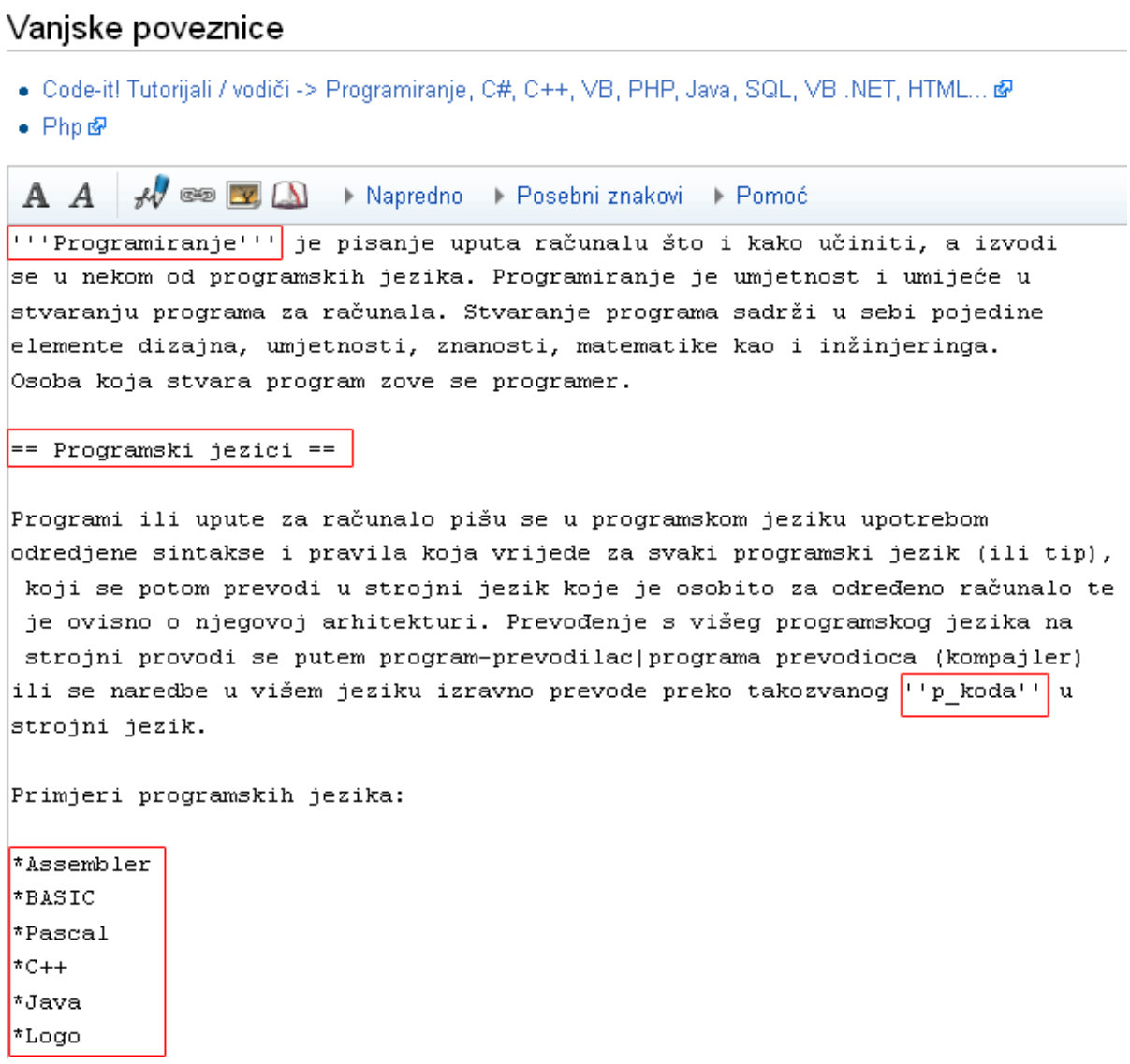
Brisanje oznaka sa WEB servera

Pored ikone povećala nalazi se mala „x“ ikona koja služi za brisanje oznaka koje se nalaze na WEB serveru.

6.12. Primjena

Jedna od mogućnosti primjene predstavljenog programa je slobodno tagiranje tekstova. Za demonstraciju tagiranja je preuzet tekst iz Wikipedije koji govori upravo o računalnom programiranju.

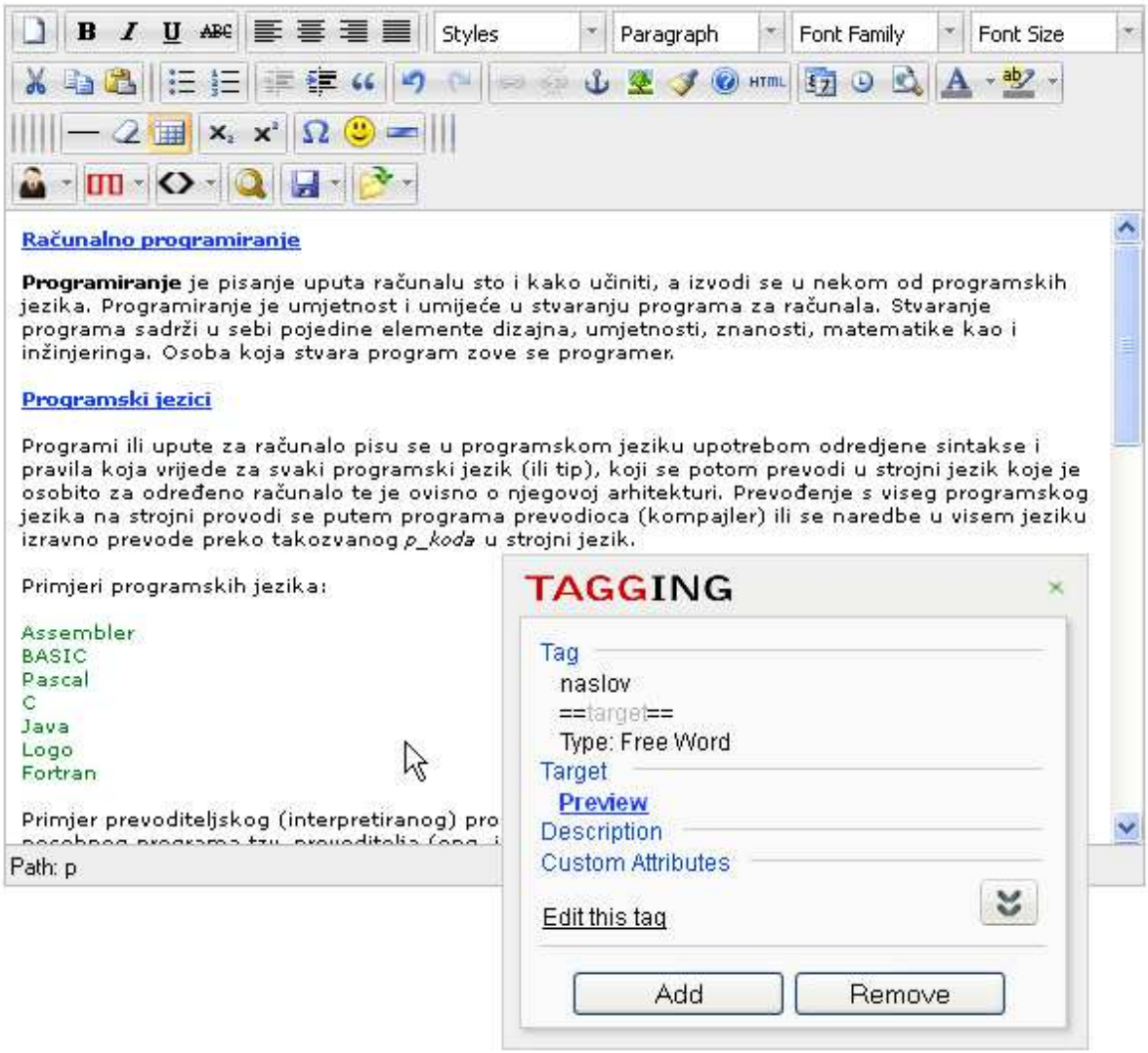
U klasičnom smislu, za kreiranje Wiki sadržaja je potrebno ručno pisati Wiki oznake kako bi algoritam koji uređuje tekstove za Wikipediju znao razlikovati naslove, podnaslove, liste, nakošena slova itd. Označen i pripremljen tekst za predaju Wikipediji izgleda kao na sljedećoj slici:



Slika 44: Wiki oznake

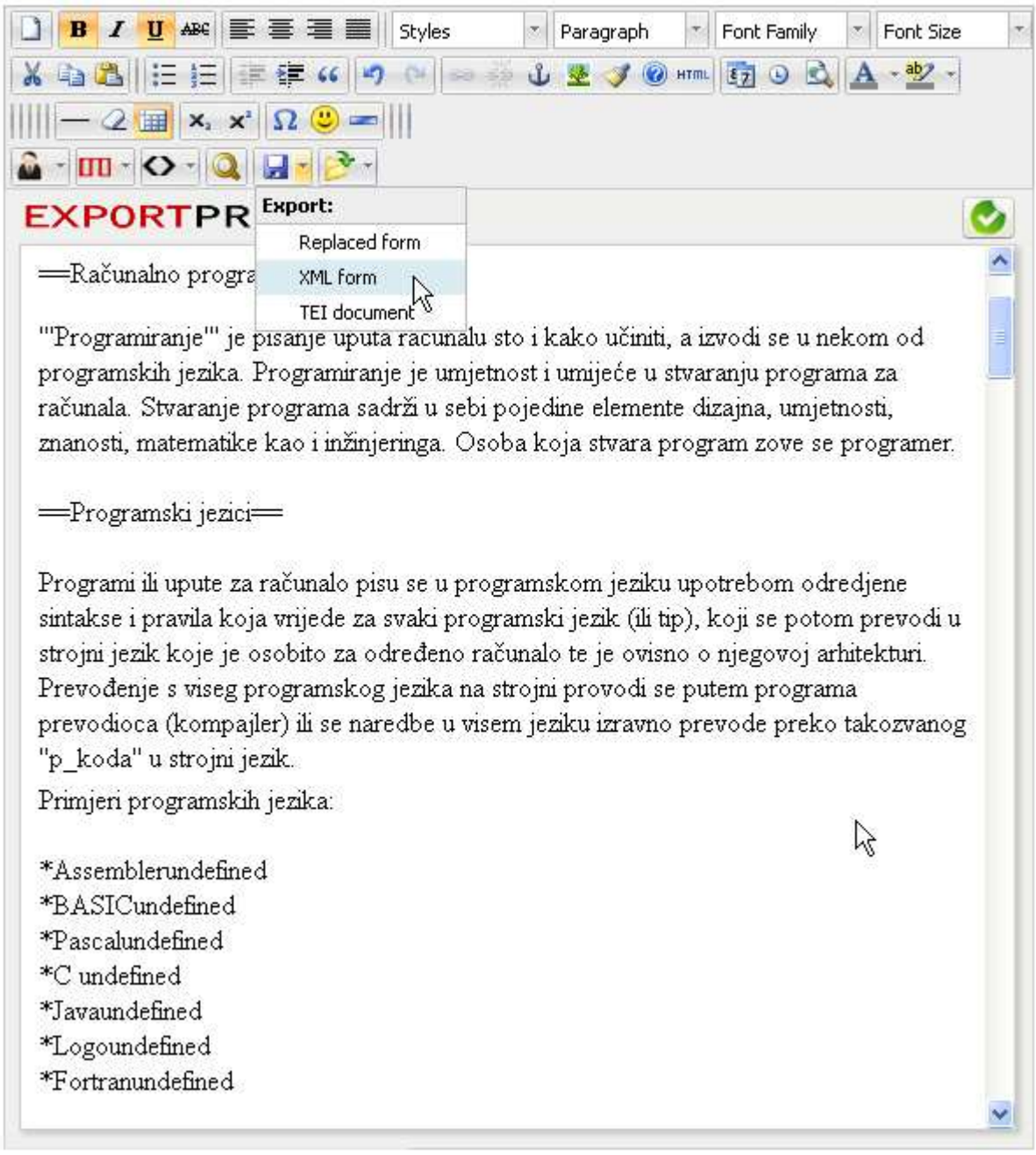
Umjesto ručnog unošenja oznaka, za označivanje Wiki teksta ćemo se koristiti TEIMark programom. Prvi korak je kreirati željene oznake unutar „New Tag“ kartice, ili ih učitati s korisničkog računa ukoliko smo ih već ranije definirali. Kreiranje oznaka je objašnjeno u poglavlju 6.3.

S izrađenim Wiki oznakama daljni postupak je daleko jednostavniji od ručnog unošenja simbola i znakova. Dovoljno je selektirati željene djelove teksta i primjeniti na njih oznake koje želimo. Označeni tekst će vizualno biti prilagođen našim željama, tj. našim postavkama unutar prethodno kreiranih tagova, kao što to prikazuje sljedeća slika:



Slika 45: Primjer oznacenog teksta

Zadnji korak koji je potrebno napraviti da bi dobili sadržaj koji odgovara Wiki tekstu je poslati zahtjev za pretvorbu teksta u nama željeni format. U ovom slučaju to je zahtjev za zamjenu teksta u XML format, kao što prikazuje sljedeća slika:



Slika 46: Primjer zamjenjenog teksta

Dobiveni tekst je samo instanca originalnog teksta i možemo ga koristiti u daljnje željene svrhe, dok je originalni tekst ostao nepromjenjen te ga je moguće nastaviti označivati s drugim oznakama, što može doći jako do izražaja ukoliko se više ljudi bavi paralelnom obradom istog teksta.

7. ZAKLJUČAK

TEIMark aplikacija predstavljena u ovom radu se sastoji od mnogih funkcija koje zbog opsežnosti nisu mogle biti opisane u sklopu ovog rada, jer izvorni kod se sastoji od 230.000 znakova tj. 7200 linija koda i ukupno 240 funkcija koje osiguravaju željenu funkcionalnost. Zbog toga se opis TEIMark aplikacije dotakao okvirno programske logike kako bi dao uvid o problematici i rješenjima koja su korištena za stvaranje ovakvog proizvoda.

Važno je i napomenuti da TEIMark aplikacija se nalazi na samom početku vlastitog razvoja i da moguće daljnje grane razvoja za različite primjene su veoma raznolike. Sam razvoj se također ne može nikako promatrati kao završan proces jer se uvijek mogu pojavljivati nova očekivanja i nove potrebe korisnika, za čije ispunjenje TEIMark ima dobru temeljnu strukturu i mogućnost relativno jednostavnog prilagođavanja tim novim potrebama.

Automatizam označivanja unutar TEIMark aplikacije, osim što ubrzava označivanje tekstova od strane korisnika, može u daljnjem razvoju biti iskorišten za nove primjene, pogotovo zbog mogućnosti ugradnje u neke druge automatske procese, čineći tako jedan složeniji automatizam. Kao potencijalna vrijednost se nameće mogućnost automatskog pretvaranja rečenica u njen strukturalni oblik, što bi se moglo iskoristiti kod naprednih automatskih WEB prevoditelja. U grubo možemo zamisliti strukturu engleske rečenice i strukturu iste Hrvatske rečenice. Jasno je da strukture nimalo ne odgovaraju jedna drugoj nego se pretvorba struktura mora provesti prema unaprijed utvrđenim pravilima za odgovarajuće jezike. Razvojem kvalitetne pretvorbe struktura rečenica iz jednog jezika u drugi, TEIMark aplikacija bi mogla postati i izvrstan prevoditelj.